CURATOR
ASSISTANT®

Software per la gestione di musei di arte contemporanea[1]

Identificativo del progetto: CA
Nome documento: Interaction design(ID)
Identificativo del documento: 8-CA_ID_E1_R2
Data del documento: 02/07/2012
Prima revisione del documento: 10/07/2012
Seconda revisione del documento: 24/07/2012

Note: Il presente documento è da allegare ai documenti già consegnati e valutati relativi al progetto di Ingegneria del Software.

1 Marchio ispirato al noto quadro di Salvador Dalí *"La persistenza della memoria – 1931"*.
Il simbolo di marchio registrato è stato usato solo a scopo simulativo di progetto reale, non ha pertanto nessuna valenza concreta.

Indice generale

Premesse

Questo documento completa i documenti (**1-CA_SDP_E1_R6** - **7-CA_TP_E1_R2**) già consegnati e valutati inerenti all'esame di 'Ingegneria del Software' e da solo ricopre la parte progettuale relativa all'esame di 'Interazione Uomo-Macchina'.

A tal proposito si simulerà di estendere l'attività di *design*, documentata in **6-CA_SD_E1_R1**, alla progettazione dell'interfaccia d'uso del progetto in esame anche se i metodi di progettazione saranno differenti da quelli visti in precedenza (si parlerà di *interaction design*).

Spesso ci si riferirà a concetti e assunzioni fatte nei documenti di 'Ingegneria del Software'. Tuttavia potranno essere riscontrate alcune incongruenze con tali documenti, questo poiché la progettazione che segue avviene dopo la discussione finale del progetto di 'Ingegneria del Software' con il docente che ha portato alla luce alcuni difetti privi di correzione perché precedenti alla consegna del progetto stesso. Le incongruenze che si possono trovare rispecchiano quindi i suggerimenti del docente rispetto all'impostazione iniziale e la rivalutazione di alcune specifiche.

Infine, si progetterà solo una parte di interfaccia utente ovvero quella relativa a ciò che nei documenti precedenti chiamavamo '*client* dedicato' (cioè l'applicazione che curatori e amministratori useranno sui loro PC per gestire il sistema) e verranno realizzate solo alcune funzioni di tale *client*. Oltre alla progettazione formale di queste parti di interfaccia si è deciso di implementarle in codice JAVA (codice che è riportato successivamente) e le schermate grafiche che verranno mostrare in questo documento sono il risultato dell'esecuzione del codice riportato. Bisogna precisare però che nonostante l'interfaccia sia stata realmente implementata essa ha sempre una valenza prototipale, sono state implementate solo le funzionalità principali in versione semplificata e limitata. Si è data particolare importanza all'interfaccia visiva e la logica dell'applicazione è rudimentale e premeditata probabilmente non esente da *bug*; la stessa interfaccia visiva è altamente semplificata ed è chiaro che una reale applicazione gestionale di un museo sia molto più complessa.

1 Introduzione

1.1 Obiettivi

Questo documento si pone l'obbiettivo di descrivere come verrà progettata l'interfaccia d'uso del sistema verso gli utenti finali. Nei documenti precedenti oggetto di attenzione è stato il sistema da progettare. Anche se lo *Unified Process* è guidato dai casi d'uso che coinvolgono inevitabilmente l'interazione tra gli utenti (attori) e il sistema, il processo di progettazione è partito dalla definizione dei *requisiti funzionali* concentrando quindi la nostra attenzione sulle funzionalità del sistema e sulla loro realizzazione. Poiché nella progettazione delle interfacce utente lo scopo è quello di progettare un sistema usabile quest'approccio non può essere continuato a seguire, bisogna porre l'attenzione in primo luogo sull'utente. Si parla quindi di *interaction design* un tipo di progettazione centrata sull'essere umano (HCD).[2]

1.2 Identificativo documento

Questo documento è identificato con il codice **8-CA_ID_E1_R2** dove E<numero> sta per l' *edizione* di numero specificato e R<numero> sta per la *revisione* di numero specificato : ad esempio E1 ed R0 sta per edizione numero 1, revisione numero 0. Per una spiegazione esaustiva dell'identificativo di documento rimandiamo al documento **1-CA_SDP** paragrafo **1.3** .

2 A tal proposito saranno seguite le linee guida del testo 'Facile da usare' di R. Pollillo.

2 Profili degli utenti

Lo scopo dell' *interaction design* è quello di progettare un sistema che si adatti all'utente. Quando si progetta un prodotto si è portati a considerare noi stessi come utenti tipici del sistema, questo è un errore perché i futuri utenti reali avranno bisogni, abitudini e preferenze diverse.

Nel documento dei casi d'uso **4-CA_UCM** si è pensato agli utenti come ruoli astratti (attori) ma il rischio è quello di perdere di vista le esigenze degli utenti reali. È molto importante individuare persone concrete, anche se fittizie, dotate di una precisa identità. Una persona non deve essere completamente inventata ma creata a partire da ricerche, come sintesi di varie caratteristiche presenti negli utenti reali che utilizzeranno il sistema. Questo ci aiuterà a considerare il prodotto progettato in modo più oggettivo.

Saranno quindi effettuate ricerche sul web e intervistati conoscenti che in qualche modo hanno a che fare con contesti simili al dominio applicativo per creare un profilo di utenti tipici del sistema che elenchi per ogni utente: formazione culturale e professionale, competenze, preferenze, comportamenti, ambiente sociale, stile di vita, caratteristiche fisiche. Tali profili aiuteranno a ricordare costantemente a chi è destinato il progetto.

2.1 Utente 1: Raffaele

Biografia, formazione culturale, formazione professionale:

Raffaele è nato a Napoli e ha 61 anni. Ottiene la laurea in 'Archeologia e storia delle arti' all'università degli studi di Napoli "Federico II" nel 1975. Dal 1976 al 1978 consegue il dottorato di ricerca in 'Storia dell'arte contemporanea' presso la stessa università. Nel 1979 inizia l'attività curatoriale collaborando all'allestimento di mostre per il Pan di Napoli e per il Museo Capodimonte di Napoli. Da allora ha lavorato da curatore *free-lance*[3], in Italia e all'estero, e nel 1995 ottiene l'incarico istituzionale come *conservatore-curatore* al M.AR.CO.R (Museo di Arte Contemporanea di Roma)[4].

3 Cioè come organizzatore di mostre indipendente slegato dalle istituzioni museali.
4 Museo fittizio che immaginiamo essere il museo di arte contemporanea che ha commissionato il progetto.

Competenze e responsabilità all'interno del museo:

Raffaele ricopre una posizione di rilievo all'interno del museo, quella di *conservatore-curatore*[5], cioè colui che è responsabile della conservazione, della sicurezza, della gestione e della valorizzazione delle collezioni a lui affidate. È responsabile, in concorso con il direttore, dell'identità e della missione del museo. Le sue competenze specifiche sono:

- programmare e coordinare le attività di inventariazione e catalogazione delle collezioni.
- partecipare ai programmi per l'incremento delle collezioni.
- condurre e coordinare attività di ricerca scientifica.
- collaborare alla valorizzazione delle collezioni attraverso le attività culturali, educative e di divulgazione scientifica.
- cura la progettazione scientifica nonché la realizzazione di mostre temporanee.
- verifica e controlla i progetti d'allestimento delle mostre temporanee.

Preferenze:

Raffaele è una persona di grande esperienza e professionalità. Da anni coordina le attività di catalogazione delle opere del museo e ha sperimentato varie metodologie per massimizzare i tempi di queste attività; è inoltre un ottimo organizzatore di mostre temporanee e cura l'allestimento delle più importanti mostre indette dal museo. Tuttavia, come si dice in gergo, svolge il suo lavoro con 'carta e penna'. Nasce in epoca *pre*-digitale e matura la sua formazione in un ambiente umanistico a quei tempi ancora lontano dalla tecnologia. Per questo possiede tuttora un atteggiamento di diffidenza e timore nei confronti degli strumenti tecnologici. Al momento della sua assunzione al M.AR.CO.R, il museo aveva da poco installato il suo primo sistema informatico per la gestione del catalogo; ma il database dal 1995 ad oggi è stato sempre interrogato dal personale istruito e Raffaele non ha fatto altro che impartire ordini. Stessa cosa nella gestione delle mostre, schizzi su foglio e telefono era quanto gli bastava per organizzare mostre di successo.

Ma in piena era digitale il museo deve adeguarsi, le informazioni sulle mostre devono essere

5 Figura professionale e relative competenze prese dalla carta nazionale delle professioni museali.

registrate e le spese sul personale istruito devono essere abbattute. I curatori devono essere in grado di gestire catalogazione e mostre con il supporto di un nuovo sistema informativo.

Il nuovo sistema informatico 'Curator assistant' promette *"un' interfaccia semplice e intuitiva per manipolare facilmente tutti i dati del database anche da utenti non specializzati"*[6]. Raffaele è preoccupato su questo aspetto vista la sua bassa dimestichezza con i sistemi informatici; è vero che probabilmente avrà dei sottoposti, curatori più giovani e più pratici, che lavoreranno per lui al computer ma saranno comunque persone con una formazione prevalentemente umanistica che dovranno imparare ad usare il sistema e cercheranno appoggio dal loro superiore. Raffaele dovrà quantomeno capire il funzionamento del nuovo sistema.

Comportamenti:

Raffaele in ambito lavorativo è esigente e severo, chiede alle persone che lavorano sotto la sua responsabilità massimo impegno e concentrazione. Se Raffaele ottiene dai suoi sottoposti quanto chiede allora ricambia con fiducia e stima, se i suoi sottoposti non lo ascoltano diventa irascibile. Se non riesce a controllare a pieno dipendenti e impegni lavorativi la sua irascibilità si ripercuote sul lavoro. E' di importanza cruciale che il nuovo sistema non innervosisca Raffaele con una sua eccessiva complessità anche espressa dai dipendenti altrimenti i danni ai reparti lavorativi possono essere seri.

Ambiente sociale:

Raffaele nasce in una famiglia benestante, suo padre chirurgo sua madre professoressa di Lettere. Da sempre è immerso in contesti culturali, dall'università a presenza attiva in svariate associazioni; le sue cerchie di conoscenze ruotano perlopiù intorno a persone che si occupano di arte. Non a caso è da 25 anni felicemente sposato con Elisabetta, restauratrice presso il Vaticano.

6 Citazione dalla proposta di contratto

Stile di vita:

Raffaele è un grande viaggiatore, inizialmente per lavoro poi per passione ha visitato i maggiori musei internazionali tra cui l' Hermitage di San Pietroburgo a cui è rimasto particolarmente affezionato. Quando Raffaele visita un museo concilia lavoro a piacere cercando di cogliere gli aspetti organizzativi degli altri musei nonché la bellezza artistica. Raffaele è anche un grande lettore, legge dai romanzi classici ai thriller moderni e ha una collezione personale di trattati d'arte. E anche appassionato di cinema classico e i suoi film preferiti sono i "Spaghetti western" di Sergio Leone. Ascolta musica Jazz. Nel tempo libero, quando non viaggia, ama stare a casa a leggere un libro o a vedere un vecchio film in compagnia della moglie.

Come abbiamo detto Raffaele usa raramente il computer, tuttavia una conoscenza minimale la possiede: è ad esempio in grado di scrivere una lettera con *word* dal computer dell'ufficio. A volte Raffaele usa il suo tempo libero per recarsi in ufficio e scrivere una lettera alla moglie.

Caratteristiche fisiche rilevanti:

Raffaele ha problemi di vista; per questo porta gli occhiali. Anche con gli occhiali a volte ha difficoltà a leggere dallo schermo di un computer. Inoltre Raffaele è affetto dalla più comune forma di daltonismo ovvero quella in cui non si riesce a distinguere il rosso dal verde.

2.2 Utente 2: Elena

Biografia, formazione culturale, formazione professionale:

Elena è nata Roma ed ha 36 anni. Nel 2001 si laurea in 'Storia dell'arte' all'università degli studi di Roma "La Sapienza". Successivamente si specializza con un master per curatori all'accademia delle belle arti di Roma. La stessa accademia nel 2003 inserisce Elena al M.AR.CO.R tramite uno *stage*. Il museo rinnoverà il contratto ad Elena nel 2004 assumendola come *catalogatore*.

Competenze e responsabilità all'interno del museo:

Il ruolo di Elena all'interno del museo è quello di *catalogatore*, cioè svolge attività d'inventariazione e catalogazione del patrimonio museale, sotto il coordinamento e la responsabilità scientifica del conservatore. Le sue competenze specifiche sono:

- partecipa alla programmazione e pianificazione delle attività di catalogazione.
- realizza le schede di inventario e di catalogo.
- contribuisce all'aggiornamento della metodologia, degli standard e degli strumenti di catalogazione adottati dal museo attraverso l'utilizzo di tecnologie informatiche e telematiche.

Preferenze:

Elena è una persona affidabile, responsabile e puntuale. Dal 2004 lavora sotto la supervisione di Raffaele alle attività di catalogazione delle opere del museo. Raffaele non si è mai interessato agli strumenti di catalogazione effettivi, la sua è un'organizzazione logica che trasmette ad Elena e colleghi. Quest'ultimi devono tradurre gli ordini di Raffaele in comandi per il personale specializzato alla gestione del database. Elena *in primis*, fa da referente a Raffaele assicurandogli che ciò che è stato ordinato, anche in modo astratto, è stato operato in modo concreto. Elena ovviamente non ha nessuna competenza in merito di interrogazioni al database ne conosce il significato di '*query SQL*', ma comunicando con il personale specializzato si è fatta un'idea (molto grossolana) di come funziona la base di dati: essa conserva un'insieme di tabelle, se ad esempio si deve aggiungere una nuova opera al catalogo verrà aggiunta una riga a qualche tabella, se vengono chieste informazioni alla base di dati essa restituisce un insieme di righe di una tabella o spesso un insieme di righe create come combinazione di più tabelle.

Adesso la situazione sta per cambiare, a breve verrà installato il nuovo sistema; Elena stessa ha proposto l'aggiornamento degli strumenti di catalogazione attraverso l'utilizzo di nuove tecnologie informatiche. Non servirà più il personale specializzato, quel poco che conosce sull'utilizzo di un computer unite alle conoscenze del suo lavoro permetteranno a lei stessa di operare gli ordini di Raffaele sulla base di dati. Elena è molto entusiasta di questo cambiamento ed è molto ottimista sui

benefici che potrà portare alla gestione del museo. Elena ad esempio si occupa anche di scrivere le schede illustrative delle opere del museo, per molti anni le ha scritte su *word* e forniva i *file* di testo ad una compagnia esterna che realizzava le schede da affiggere accanto alle opere. Il nuovo sistema si propone di fornire uno strumento che permetta a lei di creare le schede definitive. Elena spera che le sue aspettative non siano deluse da un sistema troppo complicato che neanche lei è in grado di usare correttamente.

Comportamenti:

Elena è molto veloce è efficiente sul lavoro, ma anche molto nervosa. Raffaele chiede il massimo da lei è questo fa si che lei sia sempre sotto tensione. Lei è sempre ubbidiente e non contraddice mai il suo superiore. A volte Raffaele si arrabbia facilmente e lei dissimula calma mentre dentro sente molto forte lo *stress*. Se il nuovo sistema gli provocasse ansia dovuta al fatto di non riuscire a svolgere i compiti richiesti o dovuta a situazioni di errore che non è in grado di gestire, lo *stress* potrebbe compromettere l'andamento del lavoro.

Ambiente sociale:

Elena nasce in una modesta famiglia romana, suo padre muratore sua madre casalinga . Elena *"si è fatta da sola"*, nessuno della sua famiglia ha frequentato l'università e dopo il diploma capisce che se vuole continuare gli studi deve contare solo sulle proprie forze. Alternando lavori saltuari riesce a portare avanti gli studi universitari e, negli anni in cui frequenta l'università, la facoltà di lettere e filosofia alla quale si è iscritta è ormai divenuto un ambiente multiculturale con cui scambiare principi e ideali. E' qui che Elena conosce la maggior parte delle sue amicizie, persone di diversi ceti sociali e di diversa nazionalità.

Stile di vita:

Ad Elena piace uscire; tuttora frequenta gli amici dell'università oppure organizza cene con i colleghi di lavoro nei più svariati locali della capitale. *Single* per scelta, quando Elena è a casa gli piace tenersi in contatto con i suoi amici tramite i social network, ad esempio Facebook. Non è in

grado di usare tutte le sue funzionalità ma le cose basilari riesce a farle; nonostante il suo computer di casa un po' datato riesce a destreggiarsi nella rete, gli piace ricercare notizie su ogni tipo di cosa e tenersi sempre informata. Sa poche cose sul computer ma è quanto gli basta per renderla autonoma e soddisfatta.

Tra gli altri hobby pratica *yoga*, ascolta musica *raggae* e per quanto riguarda il cinema non le interessa molto, gli unici film che gli piacciono sono le "commedie all'italiana".

Caratteristiche fisiche rilevanti:

Elena è una persona molto concentrata ma a volte soffre di forti mal di testa che non la abbandonano anche usando dei medicinali e che la distolgono dal lavoro.

2.3 Utente 3: Diego

Biografia, formazione culturale, formazione professionale:

Diego è nato a L'Aquila nel 1986. Nel 2009 ottiene la laurea triennale in 'Banca e assicurazioni' presso l'università degli studi di Roma "La Sapienza" e nel 2011 la laurea magistrale in 'Economia aziendale' presso la stessa università. Nei mesi successivi alla laurea magistrale cerca un'occupazione principalmente su internet; le offerte concrete sono poche visto il periodo di crisi che investe il Paese. Trova un annuncio relativo al M.AR.CO.R, il quale a seguito della ristrutturazione del suo sistema informativo sta cercando delle figure 'nuove' da inserire nell'area amministrativa: in questo caso specifico il museo cerca un addetto per il servizio prestiti che abbia una preparazione in materia di assicurazioni. Anche se poco convinto, Diego risponde all'annuncio. Gli viene fissato un colloquio che supera viste le sue conoscenze in campo assicurativo e viene assunto come stagista con un contratto a sei mesi.

Competenze e responsabilità all'interno del museo:

Il ruolo che dovrà ricoprire Diego all'interno del museo è quello di *registrar*, ovvero colui che assicura dal punto di vista organizzativo la movimentazione delle opere, la relativa documentazione e le procedure che la regolano, soprattutto in connessione ai prestiti. In particolare:

- redige, documenta e organizza gli atti relativi all'acquisizione, al prestito, all'assicurazione e alla spedizione delle opere.
- segue l'iter inerente al trasferimento delle opere, all'esterno e all'interno del museo.
- è responsabile delle procedure di prestito in entrata, nel caso di mostre organizzate dal museo.

Preferenze:

Diego era titubante quando ha risposto all'annuncio del museo. Le sue aspirazioni erano tutt'altre, ma in un periodo in cui trovare un lavoro non è facile ha deciso di provare. Le perplessità di Diego erano dovute al fatto di lavorare in un ambiente a lui totalmente estraneo di cui conosce ben poco, un museo. Al colloquio gli hanno spiegato che non è richiesta nessuna competenza artistico letteraria per gestire la movimentazione delle opere: ad esempio le informazioni di base delle opere in entrata al museo saranno inserite dal personale qualificato (come i *catalogatori*), lui potrà inserire in seguito i dettagli sul prestito (Data inizio prestito, Data fine prestito, Museo cedente), sarà poi suo compito stipulare un contratto con il museo cedente che dovrà essere inserito nel sistema come documento amministrativo associato all'opera in questione e preparare una polizza assicurativa il cui numero dovrà essere sempre inserito nel sistema come dato amministrativo dell'opera in entrata.

Il museo considera Diego valido e lui comprende di poter svolgere le mansioni richieste specialmente sulla parte assicurativa delle opere in entrata e in uscita. Tuttavia Diego potrà svolgere il lavoro descrittogli solo dopo l'attivazione del nuovo sistema informatico. Non gli preoccupa minimamente usare un gestionale, Diego si ritiene abbastanza pratico nell'uso del computer e conosce molto più di quello che la gente normalmente sa su di esso. Diego non ha un diploma di ragioneria come ci si potrebbe aspettare, ha un diploma di perito informatico. Dopo le superiori si accorge che la sua vocazione è per l'economia, ma non abbandona l'interesse per la tecnologia e così

impara l'uso dei più importanti gestionali di contabilità.

Diego spera che il nuovo sistema sia di qualità e non troppo banale o noioso visto che è destinato anche a persone di formazione umanistica che potrebbero non saper usare bene il computer. D'altro canto spera che sia progettato bene in modo che la parte del sistema su cui lui dovrà lavorare non gli richieda competenze artistiche che non conosce e che non gli interessano.

Comportamenti:

Diego lavora presso il museo da poco più di una settimana, il nuovo sistema ancora non è stato installato e nel frattempo gli stanno facendo vedere come venivano gestite le pratiche dei prestiti delle opere fino ad ora. Sostanzialmente tutto il lavoro era cartaceo, alcune compagnie assicurative fornivano dei *pacchetti* con cui assicurare le opere in transito e inviavano una copia cartacea della polizza al museo. Le polizze fatte fino adesso, secondo Diego, non tutelavano correttamente le opere ne il personale attuale era in grado di comprendere i numerosi cavilli legali.

Diego è una persona molto calma e pacata ma nei primi giorni di lavoro si sta annoiando perché non può utilizzare il nuovo sistema. Raffaele il suo superiore, non lo vede di buon occhio perché non ha una formazione umanistica e la pensa diversamente dal direttore del museo che ha deciso la sua assunzione. Secondo Diego, Raffaele non capisce l'importanza di avere una persona interna esperta di assicurazioni che può tutelare il museo. Tuttavia se il nuovo sistema non gli permettesse di lavorare serenamente oppure gli richiedesse competenze artistiche che non ha, la tensione tra Raffaele e Diego potrebbe aumentare e Diego sarebbe costretto a lasciare il posto.

Ambiente sociale:

Diego nasce in una famiglia agiata, suo padre è un libero professionista e sua madre lavora in una agenzia immobiliare. Ai tempi dell'università si trasferisce a Roma in una casa condivisa da studenti. Si crea un gruppo di amici ma nell'ultimo anno di università si fidanza con una ragazza di Roma e si isola dagli amici. Quando viene assunto al M.AR.CO.R. decide di affittare un nuovo appartamento con cui convivere con la ragazza.

Stile di vita:

Diego attualmente vive con la ragazza la quale sta ancora frequentando l'università, escono soltanto il fine settimana e qualche volta Diego torna all'Aquila a trovare i genitori.

Diego è appassionato di cinema e in particolare dei *collossal americani*, a volte si rivede film datati a cui è particolarmente affezionato come 'Star Wars' , 'Arancia meccanica' , 'Pulp Fiction'. Per quanto riguarda la musica non ha un genere preferito, generalmente ascolta quello che passano alla radio. È una amante degli oggetti tecnologici: possiede un telefono di ultima generazione, un *notebook* all'avanguardia e un'ultimissima console per i *videogame*; Diego è molto curioso ed esamina attentamente tutte le funzionalità dei sui oggetti tecnologici per utilizzarli al massimo delle loro potenzialità.

Caratteristiche fisiche rilevanti:

Nonostante Diego sia una persona con grandi capacità, si distrae facilmente e non riesce a rimanere concentrato per periodi molto lunghi.

3 Analisi dei compiti

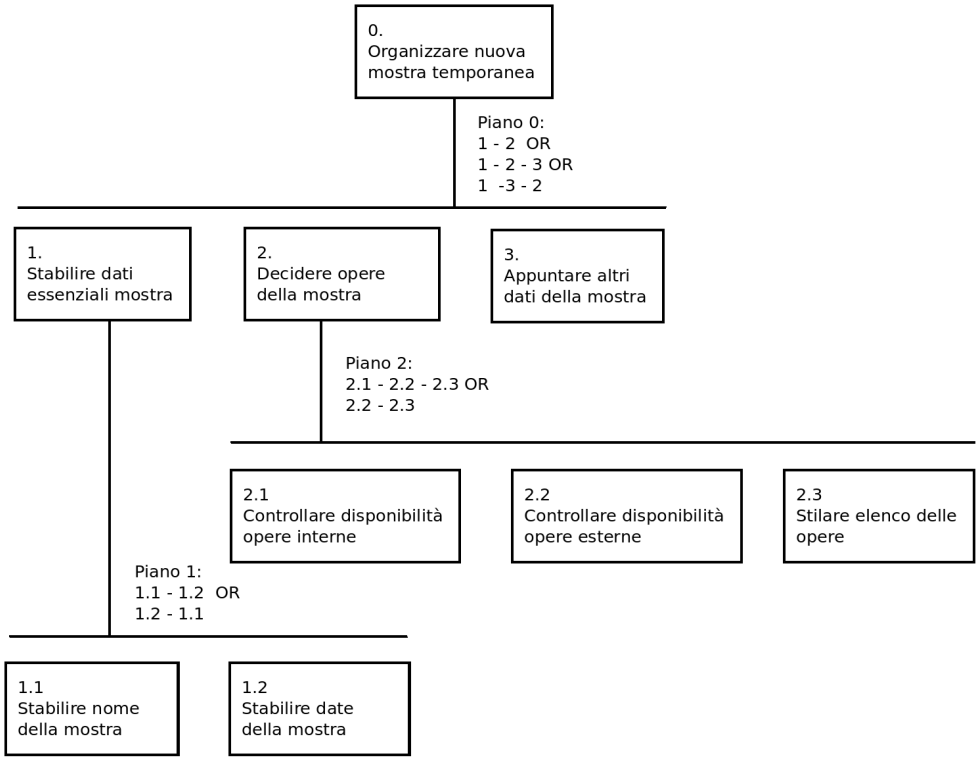
Attraverso l'analisi dei compiti si identificano le azioni necessarie ad un utente per svolgere un'attività che gli compete. Tale analisi non è una descrizione di come usare il sistema progettato, essa è preliminare alla progettazione di un'interfaccia d'uso e descrive i compiti che deve svolgere un utente per una particolare attività tralasciando l'uso del sistema stesso. Ad esempio considerando l'utente Raffaele il quale si occupa anche di organizzare mostre temporanee, esso effettuerà, per organizzare una mostra, una serie di compiti che potrà anche svolgere manualmente ovvero:

- stabilire nome e date della mostra.
- controllare la disponibilità di determinate opere durante il periodo della mostra (per quanto riguarda le opere interne della collezione permanente queste non devono essere state prestate ad un altro museo durante il periodo della mostra mentre per quanto riguarda le opere esterne della collezione temporanea i musei che prestano tali opere devono garantire disponibilità durante il periodo della mostra).
- Stilare un elenco delle opere che parteciperanno alla mostra e ritoccarlo quando si vuole.
- Appuntare dati aggiuntivi sulla mostra.

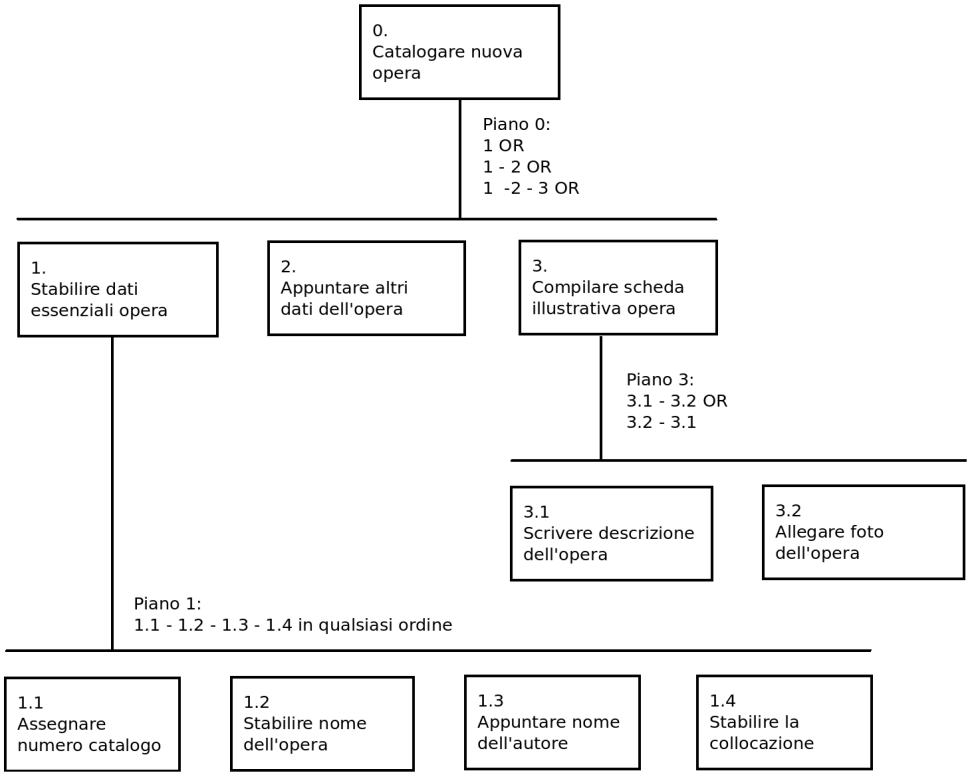
La descrizione dei compiti suggerisce l'organizzazione del menu dell'interfaccia d'uso e guidano la progettazione del dialogo. Tuttavia alcune azioni che compongono un compito potranno essere svolte in automatico dal sistema o essere lasciate allo svolgimento manuale.

Non saranno descritti tutti i compiti che può svolgere un utente nel suo ambito lavorativo ma solo quelli per cui ci interessa progettare l'interfaccia, immagineremo inoltre che un compito sia svolto da uno degli utenti di cui abbiamo creato il profilo.

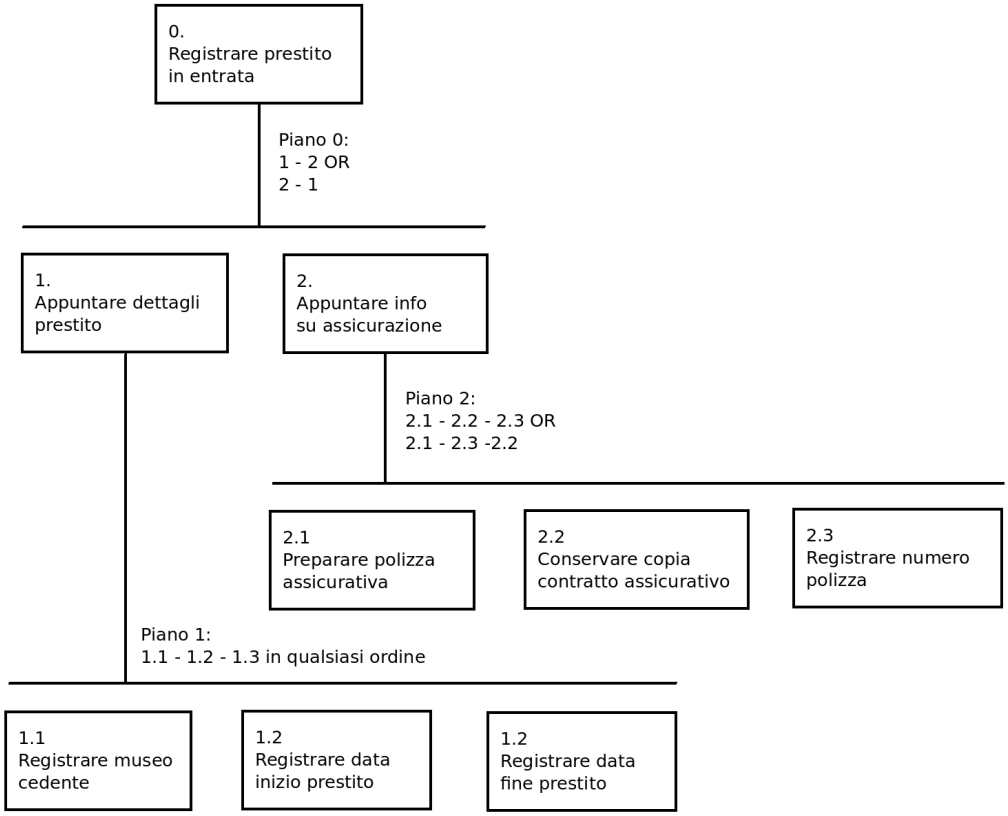
3.1 Compito A: “Organizzare nuova mostra”



3.2 Compito B: “Catalogare nuova opera”



3.3 Compito C: “Registrare prestito in entrata”



4 Scenari d'uso

Per aiutare ad immaginare un nuovo prodotto, è utile ipotizzare dei possibili scenari d'uso. Esso è una narrazione in linguaggio comune di una possibile storia dell'uso del sistema da parte di un utente tipico (uno degli utenti di cui abbiamo creato un profilo mentre svolge uno o più compiti) descritta in modo molto realistico. Infatti, l'ideazione di storie d'uso tipiche e concrete è un modo molto efficace per collocare il prodotto da progettare nei suoi possibili contesti d'uso reali e aiuta ad evidenziare requisiti centrati sull'utente che è difficile estrapolare quando si considerano soltanto le funzionalità del sistema.

Quando abbiamo analizzato il profilo degli utenti, abbiamo ipotizzato di intervistarli prima che il sistema fosse installato per cogliere aspettative e preoccupazioni. Ora immaginiamo che il sistema sia pronto all'utilizzo e che i nostri utenti tipici (Raffaele, Elena, Diego) vi comincino a lavorare.

4.1 Scenario 1: Raffaele

Sono le 10 di sera; il museo è chiuso al pubblico e i dipendenti sono andati via da qualche ora. A volte Raffaele rimane nel suo ufficio fino a tarda ora perché il silenzio del museo vuoto lo aiuta a riflettere. Sta organizzando una mostra temporanea e sta scrivendo su un foglio di carta le opere che dovranno parteciparvi: la maggior parte sono opere esterne di cui ancora deve essere richiesto il prestito ma ce ne sono alcune interne. Raffaele alza lo sguardo dal foglio e sulla seconda scrivania del suo ufficio, la scrivania dove c'è il suo computer che usa raramente, vede il monitor accesso con la schermata principale del nuovo sistema. Elena ha lasciato acceso il computer dicendo a Raffaele di provare a usarlo. Raffaele, incuriosito, vorrebbe vedere se riesce a inserire la nuova mostra da organizzare nel sistema; quindi si posiziona sulla seconda scrivania. Elena, dal momento che conosce le difficoltà di Raffaele, ha già effettuato l'accesso al sistema inserendo le sue credenziali; quindi Raffaele si trova di fronte alla pagina principale del programma di gestione (*client* dedicato). Raffaele osserva un menu ben visibile in cui risaltano le voci 'Opere' 'Mostre' 'Artisti'; clicca su 'Mostre' e ottiene l'elenco delle mostre presenti sul sistema. L'attenzione di Raffaele è catturata da un pulsante 'Inserisci nuova mostra' su cui vi clicca. Si apre una nuova finestra in cui è presente un *form* da riempire, i campi visibili sono pochi ed è specificato che sono solo i dati base della mostra

in particolare: 'Nome della mostra' , 'Data inizio mostra' e 'Data fine mostra'. Raffaele è in grado di riempire questi campi e nota che soltanto quando inserisce tutti i dati correttamente il pulsante 'Conferma' è selezionabile. Raffaele potrebbe confermare e inserire la mostra anche se questa è ancora povera di informazioni, ma nota che sotto i campi che ha riempito ci sono due voci selezionabili: 'Inserisci opere della mostra' e 'Inserisci dati opzionali'. Raffaele non vuole imbattersi in qualche problema vista la sua poca praticità quindi ignora i dati opzionali, ma visto che la fase dell'inserimento dei dati base è andata meglio di quanto si aspettava decide di provare a inserire un'opera associata alla mostra. Seleziona 'Inserisci opere della mostra' e la finestra di inserimento si espande: adesso sotto ai dati base sono visibili due riquadri, nel riquadro sinistro c'è l'elenco di tutte le opere disponibili rispetto alle date della mostra mentre il riquadro destro, etichettato con 'Opere della mostra', è vuoto. Raffaele nota il pulsante '>' etichettato a forma di freccia verso destra e intuisce che esso permette di spostare un'opera dall'elenco di tutte le opere disponibili all'elenco delle opere della mostra. Raffaele incuriosito da questo pulsante lo preme e non si accorge che in realtà non ha selezionato l'opera che vuole spostare. È il sistema che suggerisce a Raffaele di selezionare un opera dall'elenco di sinistra, quindi ripete l'operazione e l'opera selezionata compare nel riquadro destro. Raffaele comprende che adesso la mostra ha una, e sola, opera associata. Raffaele si ritiene soddisfatto e non ha intenzione di capire come inserire un'opera temporanea non ancora presente nel sistema. Erroneamente Raffaele deseleziona 'Inserisci opere della mostra' come se volesse chiudere l'espansione della finestra su cui stava lavorando ma il sistema lo avvisa che così facendo la mostra tornerà senza opere associate. Raffaele può annullare questa selezione e finalmente può cliccare su 'Conferma'. Il sistema lo avvisa che una nuova mostra è stata inserita nel sistema.

4.2 Scenario 2: Elena

Elena inizia la sua giornata lavorativa alle ore 9 del mattino, è stato da poco installato il nuovo sistema e trova sulla sua scrivania una lista cartacea di opere da catalogare tramite il programma gestionale 'Curator Assistant'. Elena controlla le mail e nota di aver ricevuto la password che, insieme al suo codice dipendente, serve per accedere al sistema. Elena quindi avvia il gestionale e gli si presenta una schermata dove inserire 'Codice dipendente' e 'Password'. Inserite le credenziali

viene visualizzata la “Pagina di benvenuto” che spiega come consultare le 'Guide veloci' e il 'Manuale d'uso” mentre in alto una barra di pulsanti. Elena nota che il primo pulsante della barra porta il nome di 'Opere' e si aspetta che tramite esso porterà avanti il suo compito; quindi premendo tale pulsante si trova di fronte ad una schermata con a sinistra la lista delle opere attualmente presenti sul sistema e a destra una zona in cui risalta il pulsante 'Inserisci opera' oltre ad una barra in cui è possibile inserire del testo etichettata con 'Filtra opere'. Elena capisce che scrivendo del testo in tale campo la lista delle opere viene filtrata a seconda del testo inserito, ciò viene confermato dal riquadro 'Suggerimenti' in basso a destra che espone proprio questo. Elena però è interessata a inserire una nuova opera e preme l'apposito pulsante. Nella nuova finestra che si apre è possibile inserire i 'Dati base' dell'opera (Nome, Autore, Collocazione) mentre al di sotto di essi trova due voci: 'Inserisci dati amministrativi dell'opera' e 'Inserisci dati opzionali dell'opera'. La prima voce non è selezionabile, Elena è al corrente che quell'opzione sarà utilizzata dal reparto di amministrazione del museo; la seconda voce è invece selezionabile. Le informazioni dell'elenco cartaceo permettono ad Elena di riempire soltanto i dati base; mentre inserisce il nome dell'autore dell'opera Elena nota che il sistema gli suggerisce dei nomi di artisti presenti nel sistema a seconda di cosa sta scrivendo, ma lei ignora tali suggerimenti sapendo il nome che deve scrivere. Nel continuare la compilazione dei dati base Elena si rende conto che il sistema gli segnala 'Artista non presente nel sistema' e che quindi l'autore che ha inserito è invalido, in effetti ha sbagliato a scrivere. Ricompila il campo autore questa volta selezionando uno dei suggerimenti del sistema e dopo aver compilato tutti i dati base correttamente gli è permesso cliccare sul tasto 'Conferma'. Il sistema la informa che una nuova opera è stata inserita e gli chiede se vuole compilare la relativa scheda illustrativa; Elena non ha informazioni sufficienti per scrivere la scheda illustrativa ma visto che è una mansione che gli spetterà vuole vedere come si presenta l'editor di scheda, quindi clicca “Si”. Le viene presentata la schermata dell'editor con la parte centrale adibita alla scrittura della descrizione dell'opera e uno spazio riservato all'inserimento dell'immagine principale. Sulla destra invece è presentato un pannello con gli strumenti classici di un editor di testi (salva, stampa, taglia, copia ecc.). Elena prova ad inserire l'immagine principale selezionando una foto da lei stessa scattata presente sul suo computer e abbozza una descrizione dell'opera. Soddisfatta chiude la scheda senza salvare.

4.3 Scenario 3: Diego

È la seconda settimana che Diego lavora al museo ed è il secondo giorno dall'installazione del nuovo sistema. Diego riceve le credenziali per accedere al sistema per e-mail; il suo è un account da amministratore ed effettua subito l'accesso. Poco dopo riceve un'altra mail con le mansioni da svolgere nella giornata, la mail spiega che il personale del reparto 'catalogatori' ha inserito delle nuove opere nel sistema, alcune di esse sono opere che verranno a breve prestate al museo e bisogna aggiungere alle informazioni base già presenti sul sistema i dettagli sul prestito e le informazioni relative all'assicurazione delle opere in questione. In un allegato della mail Diego trova la lista delle opere su cui dovrà lavorare, prende la prima opera dell'elenco e contatta telefonicamente una compagnia di assicurazioni per contrattare una polizza sull'opera. Raggiunto un accordo con la compagnia assicurativa Diego chiede alla stessa di inviargli una copia del contratto in formato PDF tramite mail. Dopo circa un quarto d'ora Diego riceve il documento PDF, lo apre, si appunta il numero della polizza e si accinge ad aggiornare le informazioni dell'opera nel sistema. Dalla "Pagina di benvenuto" clicca sul pulsante 'Opere', trova l'opera desiderata nella tabella di sinistra e fa doppio click sulla riga corrispondente per visualizzarne i dati estesi (come suggerito dallo stesso sistema). Quello che viene visualizzato a questo punto è una schermata in cui nella parte centrale ci sono per esteso tutte le informazioni dell'opera inserite precedentemente dai 'catalogatori' e nella parte di destra dei pulsanti di controllo tra cui 'Modifica opera' e 'Inserisci documento'. Diego clicca su 'Modifica opera' e si apre una finestra con in alto i campi dei 'Dati base' dell'opera, i quali non sono modificabili perché Diego ha un account da amministratore. Al di sotto dei dati base c'è la voce 'Dati amministrativi' con i campi 'Costo opera' , 'Numero polizza' e un ulteriore voce 'Inserisci dettagli prestito'. Diego seleziona proprio quest'ultima voce e la finestra si espande permettendo di definire se la relativa opera è coinvolta in un prestito in uscita o in un prestito in entrata, sceglie quindi la seconda opzione e può adesso inserire il 'Museo cedente', la 'Data di inizio prestito' e la 'Data di fine prestito'. Ora Diego inserisce il numero della polizza che si era appuntato nell'apposito campo e conferma la modifica dell'opera. Viene riportato sulla schermata dei dati estesi dell'opera in cui adesso visualizza anche i dati amministrativi da lui inseriti; quindi clicca su 'Inserisci documento' e tramite una finestra che funge da selettore di file, seleziona dal suo computer il PDF con il contratto assicurativo. Ora il documento è associato all'opera e Diego termina la prima parte del suo lavoro.

5 Progettazione, implementazione e valutazione GUI

Per rendere più scorrevole la lettura faremo vedere la progettazione di un sottoinsieme di schermate GUI del sistema e ne daremo subito implementazione e valutazione per poi passare alla progettazione del secondo sottoinsieme di schermate e così via. Le schermate saranno organizzate in base ai compiti e agli scenari visti nei due capitoli precedenti ad eccezione delle prime due schermate che rappresentano "Il contenitore principale" e le "Informazioni iniziali" le quali non si riferiscono a nessun compito o scenario in particolare.

I paragrafi che seguono seguiranno il seguente schema:

- Progettazione astratta: seguendo il metamodello UsiXML[7] progetteremo l'interfaccia utente utilizzando prima contenitori e componenti individuali astratti. Anche se la nostra interfaccia sarà grafica (come si può già intuire dagli scenari d'uso) è utile una progettazione astratta per produrre versioni dell'interfaccia differenti come ad esempio un'interfaccia vocale. Infatti partendo da una interfaccia astratta è possibile generare a partire da essa una versione dell'interfaccia vocale, una versione grafica, ecc. (ciò può essere utile ad esempio se si vuole progettare una versione del nostro sistema per utenti non vedenti)
- Progettazione concreta: sempre seguendo lo UsiXML dopo la progettazione astratta segue la progettazione concreta in cui si sceglie che tipo di interfaccia realizzare (nel nostro caso grafica) e si concretizzano i contenitori e i componenti astratti con contenitori concreti (Window, Box, Table ecc.) e componenti individuali concreti (Button, Label, MenuItem ecc.) e cioè i comuni oggetti che usano gli attuali linguaggi di programmazione OO per realizzare interfacce grafiche. Anche qui però si mantiene un certo livello di astrazione, tali oggetti infatti sono di tipo generico (ad esempio il contenitore generico Box corrisponde al JPanel delle librerie *Java Swing*) questo serve a produrre versioni grafiche diverse della stessa interfaccia (se si vuole produrre una versione C++ ad esempio non avremo lo specifico contenitore JPanel ma avremo un contenitore Box in generale).
- Implementazione fisica: la relazione tra progettazione concreta e l'implementazione in codice è quasi 1:1, si tratta di tradurre il linguaggio di progettazione nel linguaggio di programmazione scelto. Nel nostro caso è stato scelto di programmare realmente

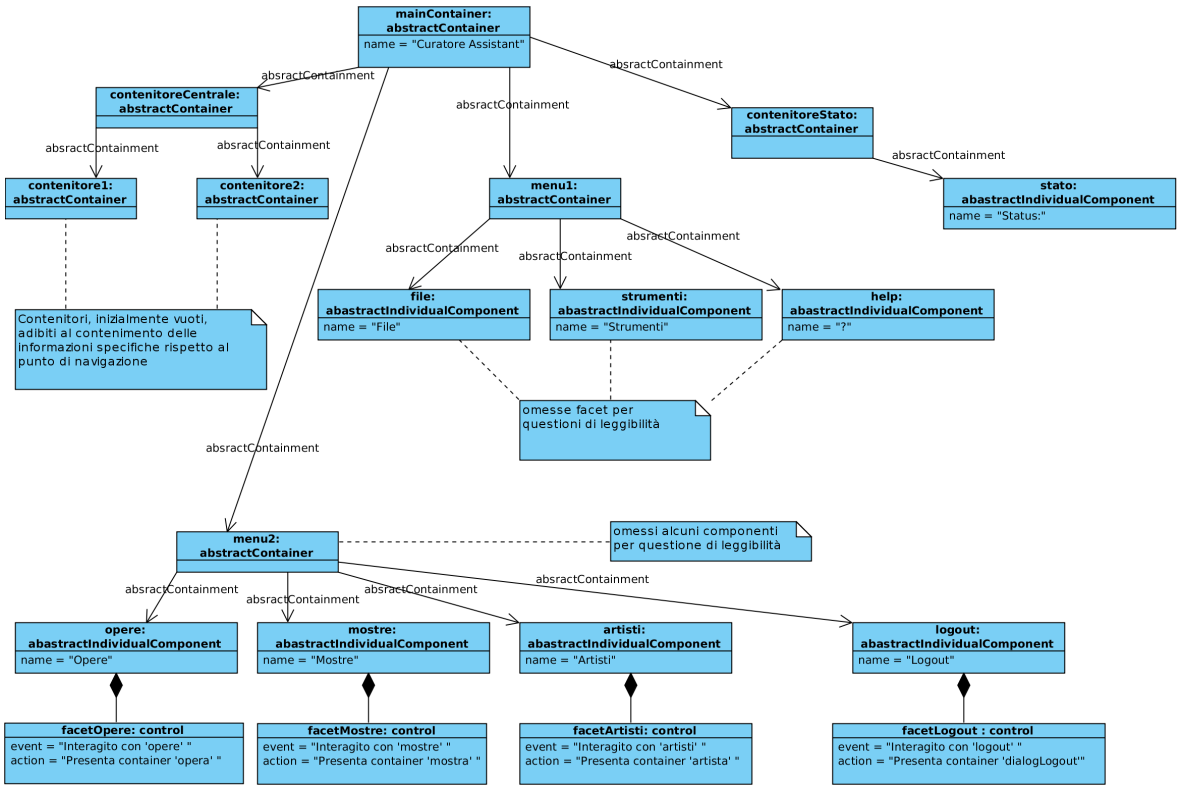
7 Linguaggio di markup basato su XML per la definizione di interfacce utente

l'interfaccia utente utilizzando Java e le librerie javax.swing.* . Quindi verrà riportato il codice che riguarda le schermate in esame.

- GUI: verranno riportati dei *screenshot* delle schermate dell'interfaccia grafica ottenute mandando in esecuzione il codice scritto.
- Valutazione: le schermate riportate verranno valutate e testate attraverso le *euristiche di Nielsen*, verificando la conformità ai principi e alle linee guida proposti dallo stesso Nielsen per valutare l'usabilità dell'interfaccia. La valutazione euristica non può sostituirsi alla valutazione con coinvolgimento dell'utente[8] che permette di portare alla luce problemi legati all'interfaccia che le euristiche non riescono a catturare. Queste attività di valutazione sono molto importanti perché, visto che la progettazione e l'implementazione sono fasi molto tecniche, non bisogna dimenticare che si sta *"progettando per l'utente"* e bisogna fare in modo che si risolvano eventuali difetti nell'usabilità del sistema.

5.1 "Contenitore principale"

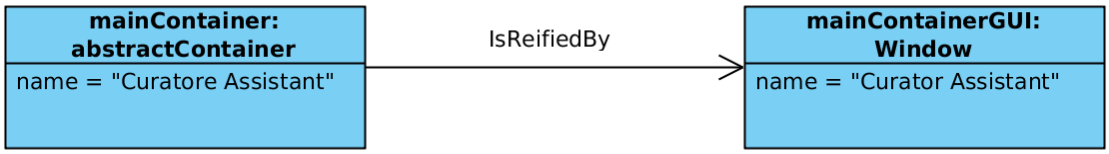
5.1.1 Progettazione astratta "Contenitore principale"



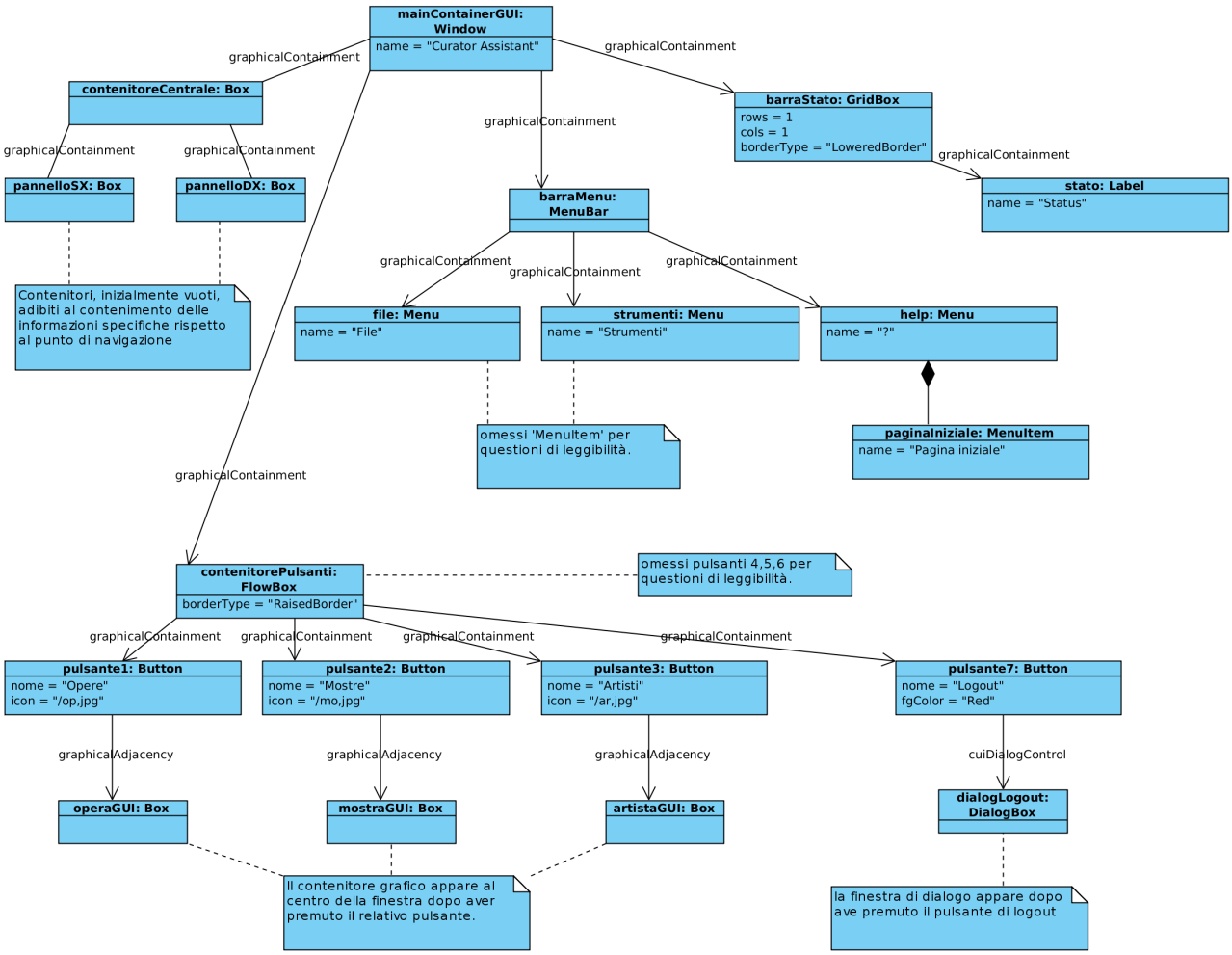
8 Che nel nostro caso corrisponde alla revisione dell'interfaccia da parte del docente.

5.1.2 Progettazione concreta “Contenitore principale”

Progetteremo concretamente l'interfaccia grafica del “Contenitore principale” cioè la parte di interfaccia che verrà visualizzata nella maggior parte delle schermate del sistema e che farà da contorno. Nel passare dalla progettazione astratta alla progettazione concreta andrebbe rappresentato l'oggetto astratto (contenitore o componente) messo in relazione con l'oggetto concreto con cui l'oggetto astratto è stato concretizzato, in questo modo:



Ma per non appesantire il diagramma di progettazione astratta con le relazioni 'IsReifiedBy' verrà proposto il diagramma con i soli oggetti di interazione concreti e confrontando questo con il diagramma astratto si capiranno quali relazioni 'isReifiedBy' sono state operate.



5.1.3 Implementazione fisica “Contenitore principale”

```
package curatorAssistantGUI_betaVs;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class MainContainerGUI extends JFrame {

        protected Container pannelloSX;
        protected Container pannelloDX;
//---------------------------Frammento omesso 1----------------------------------

//-------------------------------------------------------------------------------

        public MainContainerGUI(){

                super("Curator Assistant");

                JMenuBar barraMenu = new JMenuBar();
                JMenu file = new JMenu("File");
                JMenu strumenti = new JMenu("Strumenti");
                JMenu help = new JMenu("?");
                JMenuItem paginaIniziale = new JMenuItem("Pagina iniziale");

                help.add(paginaIniziale);
                barraMenu.add(file);
                barraMenu.add(strumenti);
                barraMenu.add(help);

                Image i=java.awt.Toolkit.getDefaultToolkit().getImage(
               "/home/macash/Desktop/CollegamentoIUM/img/op.jpg");
                i=i.getScaledInstance(40,40,Image.SCALE_DEFAULT);
                ImageIcon ic = new ImageIcon(i);

                Image i2=java.awt.Toolkit.getDefaultToolkit().getImage(
               "/home/macash/Desktop/CollegamentoIUM/img/mo.jpg");
                i2=i2.getScaledInstance(40,40,Image.SCALE_DEFAULT);
                ImageIcon ic2 = new ImageIcon(i2);

                Image i3=java.awt.Toolkit.getDefaultToolkit().getImage(
               "/home/macash/Desktop/CollegamentoIUM/img/ar.jpg");
                i3=i3.getScaledInstance(40,40,Image.SCALE_DEFAULT);
                ImageIcon ic3 = new ImageIcon(i3);

                Image i4=java.awt.Toolkit.getDefaultToolkit().getImage(
               "/home/macash/Desktop/CollegamentoIUM/img/ric.jpg");
                i4=i4.getScaledInstance(40,40,Image.SCALE_DEFAULT);
                ImageIcon ic4 = new ImageIcon(i4);

                Image i5=java.awt.Toolkit.getDefaultToolkit().getImage(
               "/home/macash/Desktop/CollegamentoIUM/img/pub.jpg");
                i5=i5.getScaledInstance(40,40,Image.SCALE_DEFAULT);
                ImageIcon ic5 = new ImageIcon(i5);

                Image i6=java.awt.Toolkit.getDefaultToolkit().getImage(
               "/home/macash/Desktop/CollegamentoIUM/img/doc.jpg");
                i6=i6.getScaledInstance(40,40,Image.SCALE_DEFAULT);
                ImageIcon ic6 = new ImageIcon(i6);

                final JFrame temp = this;

                JButton pulsante1= new JButton("Opere");
                pulsante1.setIcon(ic);

                JButton pulsante2= new JButton("Mostre");
                pulsante2.setIcon(ic2);

                JButton pulsante3= new JButton("Artisti");
                pulsante3.setIcon(ic3);

                JButton pulsante4= new JButton("Ricerca");
                pulsante4.setIcon(ic4);

                JButton pulsante5= new JButton("Pubblica");
                pulsante5.setIcon(ic5);
```

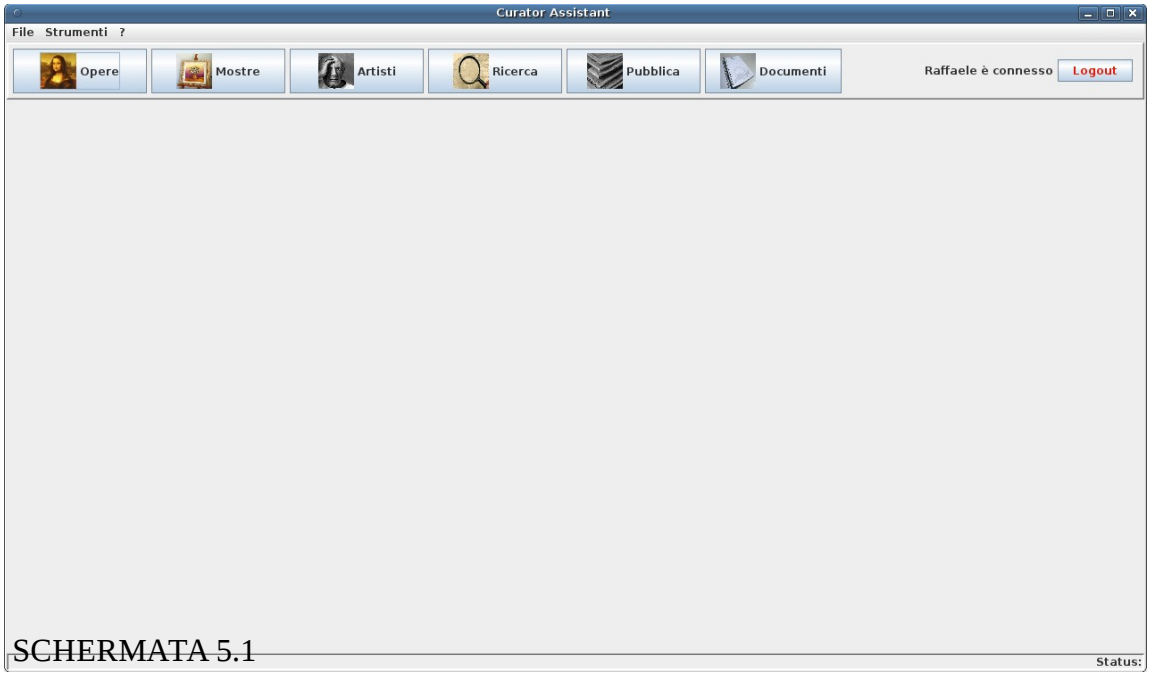
```
                JButton pulsante6= new JButton("Documenti");
                pulsante6.setIcon(ic6);

                JButton pulsante7= new JButton("Logout");
                pulsante7.setForeground(new Color(231,25,6));

                pulsante7.addActionListener(new ActionListener() {

                        public void actionPerformed(ActionEvent e) {

                                int risp = JOptionPane.showConfirmDialog(temp,
                         "Vuoi veramente disconnetterti dal sistema?",
                         "Conferma Disconnessione",JOptionPane.YES_NO_OPTION);
                                if (risp==JOptionPane.YES_OPTION) {temp.dispose();}

                        }

               });

                JPanel contenitorePulsanti = new JPanel();
               String s = new String(" "+" "+" "+" "+" "+" "+" "+" "+" "+" "+" "+" "+" "+" "+" "+" "+" ");
               JLabel la = new JLabel(s+"Raffaele è connesso");
                Dimension d = new Dimension();
                d.setSize(150, 50);
                pulsante1.setPreferredSize(d);
                pulsante2.setPreferredSize(d);
                pulsante3.setPreferredSize(d);
                pulsante4.setPreferredSize(d);
                pulsante5.setPreferredSize(d);

                contenitorePulsanti.add(pulsante1);
                contenitorePulsanti.add(pulsante2);
                contenitorePulsanti.add(pulsante3);
                contenitorePulsanti.add(pulsante4);
                contenitorePulsanti.add(pulsante5);
                contenitorePulsanti.add(pulsante6);
                contenitorePulsanti.add(la);
                contenitorePulsanti.add(pulsante7);
                contenitorePulsanti.setBorder(BorderFactory.createRaisedBevelBorder());

                LayoutManager ly = new FlowLayout(FlowLayout.LEFT);
                contenitorePulsanti.setLayout(ly);

                JLabel stato = new JLabel("Status:", JLabel.TRAILING);
                JPanel barraStato = new JPanel(new GridLayout(1, 1));
                barraStato.setBorder(BorderFactory.createLoweredBevelBorder());
                barraStato.add(stato);

                this.setJMenuBar(barraMenu);
                this.add(contenitorePulsanti, BorderLayout.NORTH);
                this.add(barraStato,BorderLayout.SOUTH);
//--------------------------Frammento omesso 2----------------------------------

//------------------------------------------------------------------------------
                this.pack();
                this.setSize(3000,3000);
                this.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
                this.addWindowListener(new WindowAdapter() {

                        @Override public void windowClosing(WindowEvent e) {

                      int answer = JOptionPane.showConfirmDialog(
                                      e.getWindow(),
                        "Vuoi veramente chiudere la finestra? \n\nChiudendo rimarrai comunque connesso",
                                      "Conferma chiusura",
                                      JOptionPane.YES_NO_OPTION);
                                                if(answer == JOptionPane.YES_OPTION) {
                                                        e.getWindow().dispose();
                                                }
                        }

                });
//--------------------------Frammento omesso 3----------------------------------

//------------------------------------------------------------------------------
               this.setVisible(true);

        }

}
```

Per far funzionare la classe precedente usare la seguente classe 'Main' e far partire da essa l'esecuzione, affinché siano correttamente caricate le immagini seguire le istruzioni nel file .zip contenente il progetto:

```
package curatorAssistantGUI_betaVs;


import javax.swing.JFrame;
import java.awt.EventQueue;
public class Main {
        public static void main(String[] args) {

                class Starter implements Runnable {
                                public void run() {

                                        new MainContainerGUI();

                                }

                }
        Runnable task = new Starter();
        EventQueue.invokeLater(task);

        }

}
```

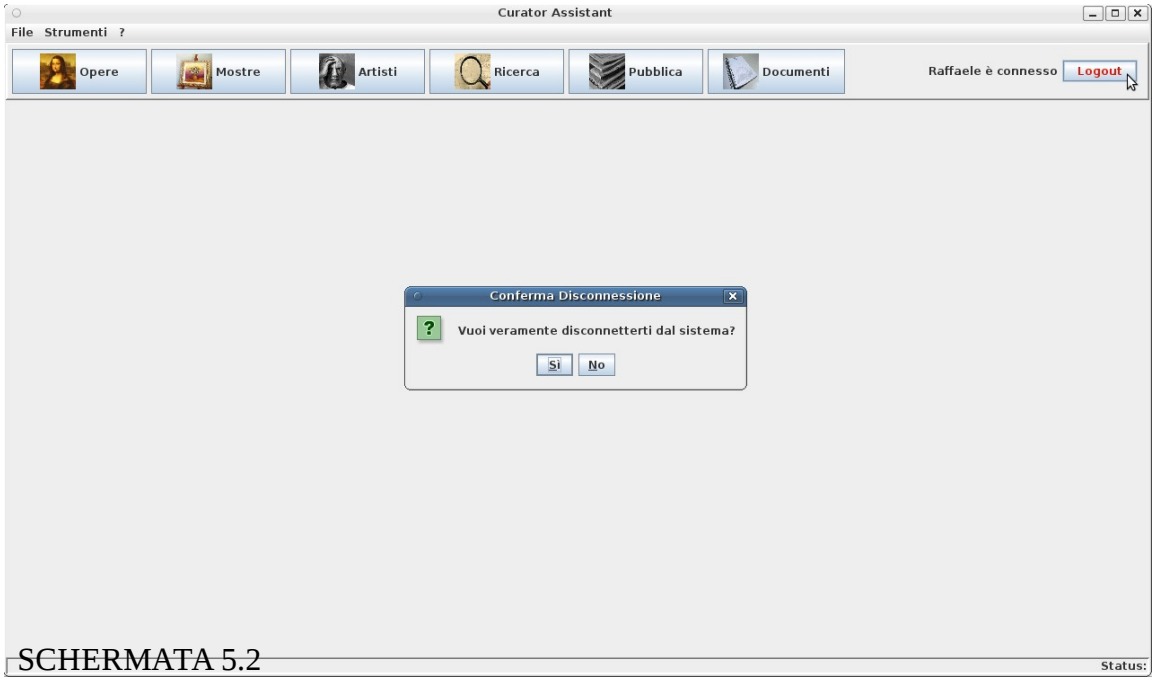
5.1.4 GUI “Contenitore principale”

Mandando in esecuzione il codice precedente otteniamo la seguente schermata. La parte centrale della schermata rimane vuota perché li andranno le informazioni specifiche rispetto al punto di navigazione in cui si trova l'utente all'interno dell'interfaccia.



SCHERMATA 5.1

Se si clicca su 'Logout', viene mostrata la seguente *dialog* :



SCHERMATA 5.2

5.1.5 Valutazione “Contenitore principale”

Visibilità dello stato del sistema:

La barra di stato in basso è stata inserita appositamente per fornire sempre una descrizione di cosa sta facendo il sistema.

Corrispondenza fra il mondo reale e il sistema:

Il menu di pulsanti ben visibili parla il linguaggio dell'utente, infatti in primo piano troviamo i pulsanti 'Opere', 'Mostre', 'Artisti' che si riferiscono a concetti molto familiari all'utente tipico del sistema. La maggior parte dei compiti riguardano opere, mostre, e artisti e l'utente può subito indirizzarsi nell'attività desiderata cliccando su uno di questi pulsanti.

Libertà e controllo da parte degli utenti:

Il menu dei pulsanti sarà sempre visualizzato durante la navigazione nell'interfaccia, pertanto se un utente preme per sbaglio un pulsante non desiderato esso può selezionare successivamente il pulsante corretto in tutta libertà visto che la barra dei pulsanti rimarrà nella stessa posizione.

Consistenza e standard:

Il layout del “Contenitore principale” sarà sempre coerente in tutte le schermate inoltre è rispettata la convenzione di avere, oltre alla barra dei pulsanti, la *menubar* (File, Strumenti ecc.) tipica di tutti i programmi a finestre. Il menu '?' rappresenta il menu *help* come convenzione. Le voci interne della *menubar* non replicheranno le funzioni offerte dalla barra dei pulsanti per non porre ambiguità all'utente, esse consentiranno invece l'accesso alle opzioni di settaggio del programma disponibili agli utenti più esperti ed è per questo che la *menubar* è meno in risalto rispetto alla barra dei pulsanti.

Prevenzione degli errori:

I pulsanti hanno una dimensione abbastanza grande per evitare che l'utente prema accidentalmente il pulsante sbagliato. La *menubar*, che contiene funzioni di settaggio più complesse, è meno visibile e non cattura l'attenzione dell'utente meno esperto che andrà ad operare sui pulsanti. Il pulsante di 'Logout' ha la scritta di colore rosso, colore che nella convenzione occidentale viene associato allo STOP, quindi gli utenti sono inconsciamente allontanati dall'effettuare il *logout*, azione che di fatto creerebbe noie all'utente se effettuata per sbaglio perché dovrebbero accedere di nuovo al sistema fornendo le loro credenziali. Da notare che l'informazione di colore rosso non è un'informazione fondamentale, non bisogna dimenticarsi che l'utente Raffaele essendo daltonico potrebbe vedere la scritta 'Logout' di colore nero. Questo non è un grosso problema, il pulsante 'Logout' è comunque differenziato dagli altri, più piccolo, più distanziato e senza immagine e anche Raffaele è inconsciamente portato ad evitarlo rispetto agli altri pulsanti. Il pulsante 'Logout' è comunque protetto da una messaggio di conferma da parte dell'utente se esso viene premuto per sbaglio (come è possibile vedere dalle immagini).

Riconoscere piuttosto che ricordare:

Come abbiamo detto i pulsanti di interazione fondamentali sono ben visibili e l'utente non deve ricordare come svolgere le operazioni più frequenti.

Flessibilità ed efficienza d'uso:

Utilizzando la *menubar*, quasi invisibile all'utente novizio, l'utente esperto può personalizzare alcune opzioni del programma che renderanno più veloce la sua interazione.

Design minimalista:

Il numero dei pulsanti è minimale, sono stati selezionati quelli per l'accesso alle operazioni più frequenti e sono state evitate informazioni irrilevanti e necessarie solo di rado. Queste informazioni verranno nascoste nella *menubar* per non togliere attenzione all'utente e comunque anche il numero

di menu nella *menubar* è ridotto ai minimi termini.

Aiutare gli utenti a riconoscere gli errori e correggerli:

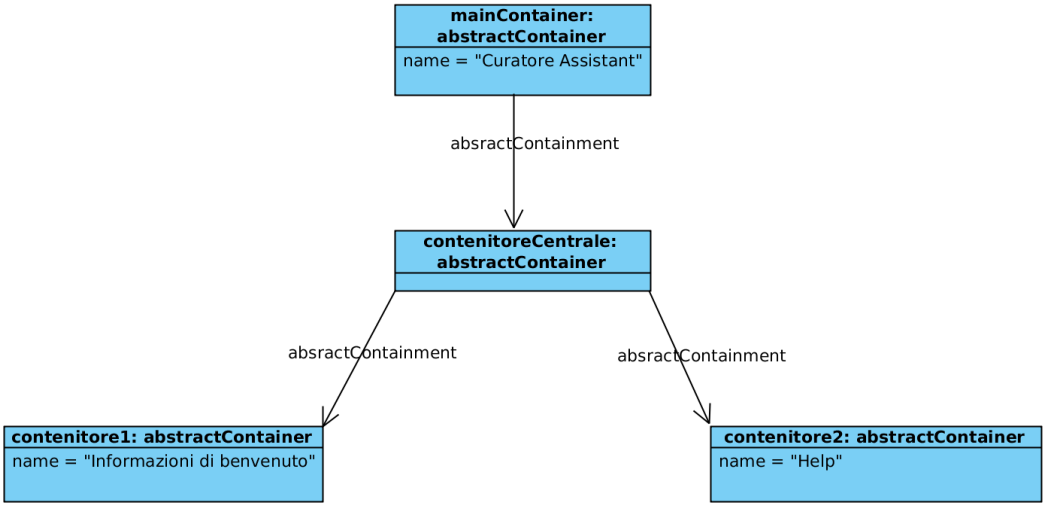
I messaggi di errore sono espressi in linguaggio molto semplice, come è possibile vedere dalla finestra di dialogo che appare quando un utente preme il pulsante 'Logout' nell'immagine precedente. L'utente può correggere l'eventuale errore di pressione accidentale del pulsante 'Logout' con una semplice scelta.

Guida e documentazione:

Il menu *help*, denotato come di consueto con il simbolo '?', è sempre accessibile qualsiasi sia il punto di navigazione nell'interfaccia perché appartenente al "Contenitore principale" che fa da contorno alla maggior parte delle schermate grafiche. Gli utenti di solito evitano di leggere le guide e provano per tentativi, quindi è stato ritenuto inutile creare un pulsante 'Help' che occupasse dello spazio prezioso e si è pensato più opportuno inserire l'*help* nella *menubar* cosicché l'utente vi acceda quando realmente vuole.

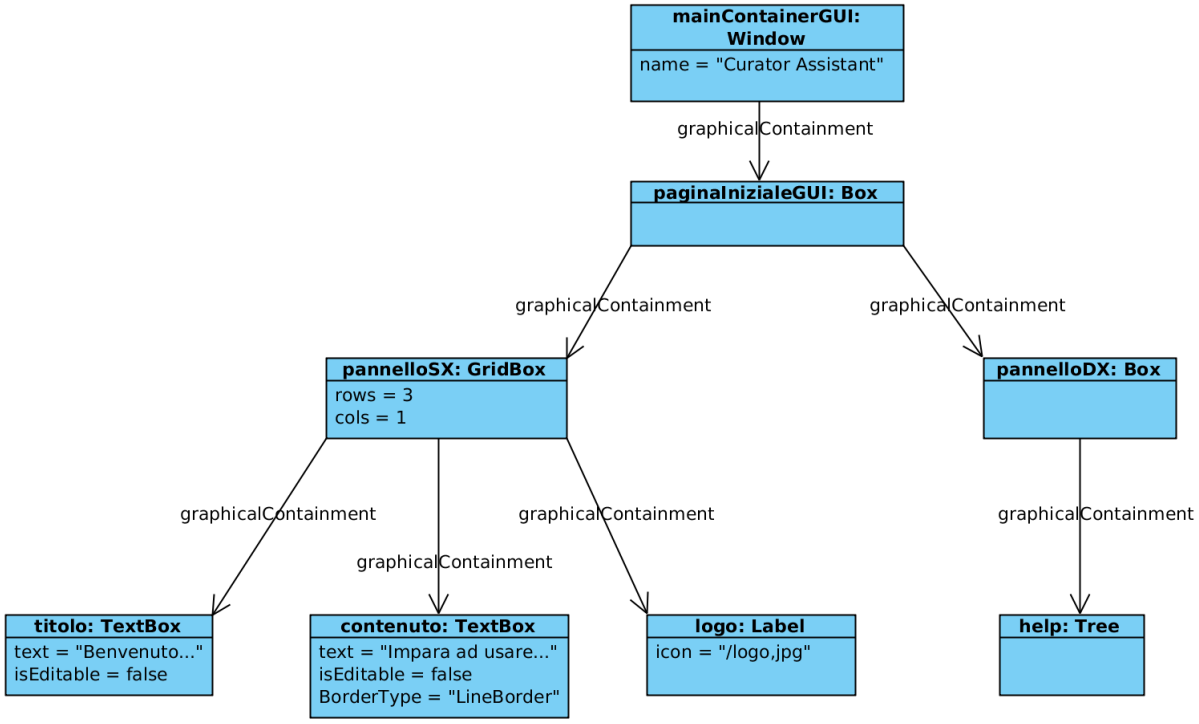
5.2 *"Informazioni iniziali"*

5.2.1 Progettazione astratta "Informazioni iniziali"



Il “Contenitore principale” come mostrato nel paragrafo precedente, contiene due contenitori adibiti al contenimento delle informazioni specifiche rispetto al punto di navigazione in cui ci si trova nell'interfaccia. In generale si ha un “contenitore 1” che elenca delle informazioni e un “contenitore 2” che consente il controllo da parte dell'utente sulle informazioni, anche se lo schema logico potrebbe leggermente variare a seconda dei casi specifici. Questo vale anche per le “Informazioni iniziali” (ovvero le informazioni che il sistema da all'utente appena effettuato l'accesso), vediamone la progettazione concreta:

5.2.2 Progettazione concreta “Informazioni iniziali”



In questa parte dell'interfaccia, 'paginaInizialeGUI' è la concretizzazione del 'contenitoreCentrale' e 'pannelloSX' e 'pannelloDX' sono la concretizzazione di “contenitore1” e “contenitore2”.

5.2.3 Implementazione fisica “Informazioni iniziali”

Per visualizzare la schermata iniziale completare il 'Frammento omesso 2' del codice precedente con il seguente frammento:

```
final PaginaInizialeGUI panPaginaIniziale = new PaginaInizialeGUI(this);
this.add(pannelloSX, BorderLayout.WEST);
this.add(pannelloDX,BorderLayout.EAST);
```

Completare il 'Frammento omesso 3' con:

```
paginaIniziale.addActionListener(new ActionListener() {

                        public void actionPerformed(ActionEvent e){


                                if(pannelloSX != null && pannelloDX != null){
                                        temp.remove(pannelloSX);
                                        temp.remove(pannelloDX);
                                }


                                panPaginaIniziale.ridisegna();
                                temp.add(pannelloSX, BorderLayout.WEST);
                               temp.add(pannelloDX,BorderLayout.EAST);
                                temp.repaint();

                        }

                }});
```

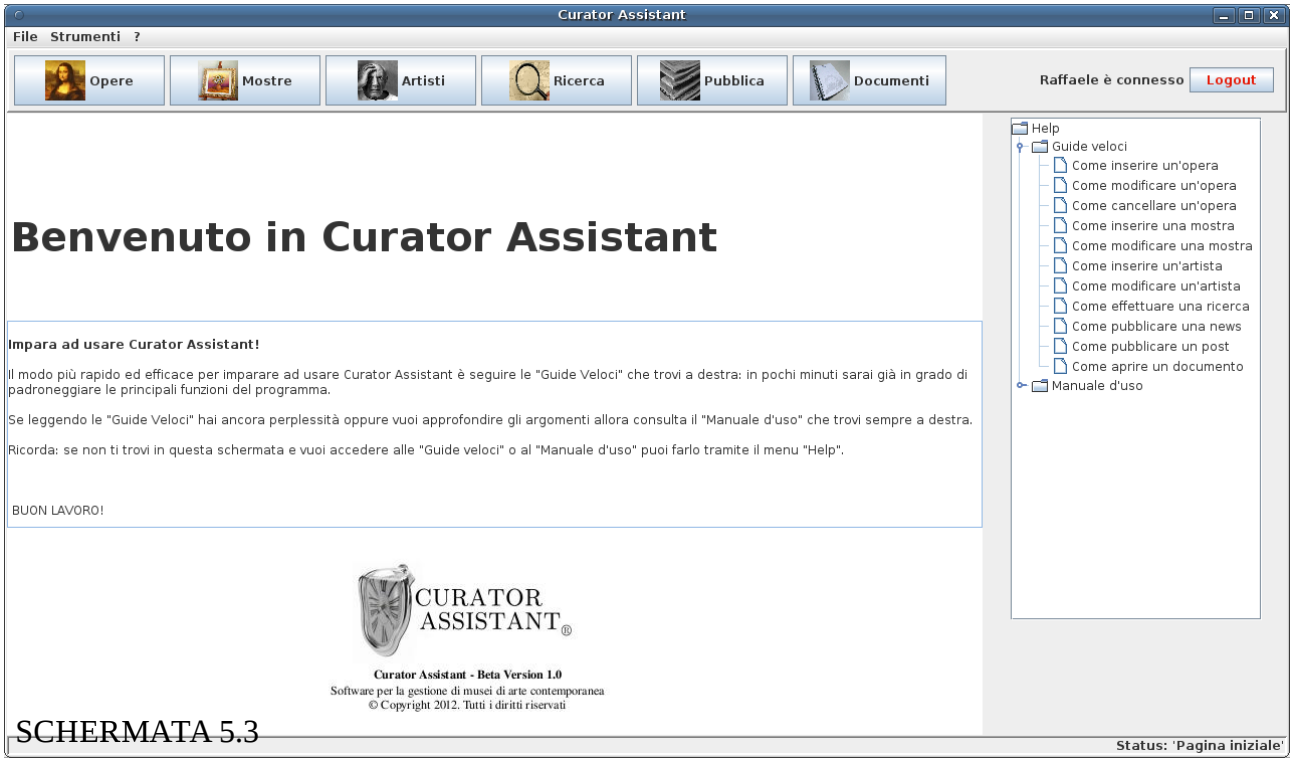
Aggiungere al package la classe 'PaginaInizialeGUI':

```
package curatorAssistantGUI_betaVs;
import javax.swing.*;
import java.awt.*;
import javax.swing.text.*;
import javax.swing.tree.*;

public class PaginaInizialeGUI {

        JPanel panSX, panDX;
        MainContainerGUI menuPrincipale;

public PaginaInizialeGUI(MainContainerGUI fin){

                panSX = new JPanel();
                panDX = new JPanel();
                menuPrincipale=fin;

                LayoutManager ly2 = new GridLayout(3,1);
                panSX.setLayout(ly2);

                JTextPane titolo = new JTextPane();
                JTextPane contenuto = new JTextPane();

                titolo.setEditable(false);
                titolo.setFont(new Font(titolo.getFont().getName(), titolo.getFont().getStyle(), 40 ));

                SimpleAttributeSet gras = new SimpleAttributeSet();
                StyleConstants.setBold(gras, true);
                titolo.setCharacterAttributes(gras, true);
                titolo.setText("\n\nBenvenuto in Curator Assistant");
                titolo.setPreferredSize(new Dimension(950,300));
```

```
                contenuto.setCharacterAttributes(gras, true);
                contenuto.setText("\nImpara ad usare Curator Assistant!");
                StyleConstants.setBold(gras, false);
                contenuto.setCharacterAttributes(gras, false);
String s1 = new String("\n\nIl modo più rapido ed efficace per imparare ad usare Curator ");
String s2 = new String("Assistant è seguire le \"Guide Veloci\" che trovi a destra: in pochi ");
String s3 = new String("minuti sarai già in grado di\npadroneggiare le principali funzioni del programma. ");
String s4 = new String("\n\nSe leggendo le \"Guide Veloci\" hai ancora perplessità oppure vuoi approfondire ");
String s5 = new String("gli argomenti allora consulta il \"Manuale d'uso\" che trovi sempre a destra.");
String s6 = new String("\n\nRicorda: se non ti trovi in questa schermata e vuoi accedere alle");
String s7 = new String(" \"Guide veloci\" o al \"Manuale d'uso\" puoi farlo tramite il menu \"Help\".");
contenuto.replaceSelection(s1+s2+s3+s4+s5+s6+s7+" \n\n\n\n BUON LAVORO! ");
                contenuto.setEditable(false);
                contenuto.setBorder(BorderFactory.createLineBorder(new Color(146,185,230)));

Image logo=java.awt.Toolkit.getDefaultToolkit().getImage("/home/macash/Desktop/CollegamentoIUM/img/logo3.jpg");
                ImageIcon icLogo = new ImageIcon(logo);
                JLabel etLogo = new JLabel();
                etLogo.setIcon(icLogo);


                panSX.add(titolo);
                panSX.add(contenuto);
                panSX.add(etLogo);

                //JTree
                panDX.setPreferredSize(new Dimension(300,100));
                DefaultMutableTreeNode root= new DefaultMutableTreeNode("Help");
                DefaultMutableTreeNode subroot1= new DefaultMutableTreeNode("Guide veloci");
                DefaultMutableTreeNode subroot2= new DefaultMutableTreeNode("Manuale d'uso");
                DefaultMutableTreeNode child= new DefaultMutableTreeNode("Come inserire un'opera");
                DefaultMutableTreeNode ch= new DefaultMutableTreeNode("Come modificare un'opera");
                DefaultMutableTreeNode ch1= new DefaultMutableTreeNode("Come cancellare un'opera");
                DefaultMutableTreeNode child2= new DefaultMutableTreeNode("Come inserire una mostra");
                DefaultMutableTreeNode ch3= new DefaultMutableTreeNode("Come modificare una mostra");
                DefaultMutableTreeNode child3= new DefaultMutableTreeNode("Come inserire un'artista");
                DefaultMutableTreeNode ch4= new DefaultMutableTreeNode("Come modificare un'artista");
                DefaultMutableTreeNode child4= new DefaultMutableTreeNode("Come effettuare una ricerca");
                DefaultMutableTreeNode child5= new DefaultMutableTreeNode("Come pubblicare una news");
                DefaultMutableTreeNode child6= new DefaultMutableTreeNode("Come pubblicare un post");
                DefaultMutableTreeNode child7= new DefaultMutableTreeNode("Come aprire un documento");
                root.add(subroot1);
                root.add(subroot2);
                subroot1.add(child);
                subroot1.add(ch);
                subroot1.add(ch1);
                subroot1.add(child2);
                subroot1.add(ch3);
                subroot1.add(child3);
                subroot1.add(ch4);
                subroot1.add(child4);
                subroot1.add(child5);
                subroot1.add(child6);
                subroot1.add(child7);
                DefaultMutableTreeNode child11= new DefaultMutableTreeNode("Indice");
                subroot2.add(child11);
                DefaultTreeModel mod =new DefaultTreeModel(root);
                JTree tree = new JTree(mod);
                JScrollPane sp = new JScrollPane(tree);
                sp.setPreferredSize(new Dimension(250,500));

                panDX.add(sp);

                fin.pannelloSX=panSX;
                fin.pannelloDX=panDX;

        }

        public void ridisegna(){

                menuPrincipale.pannelloSX=panSX;
                menuPrincipale.pannelloDX=panDX;

        };


}
```

5.2.4 GUI “Informazioni iniziali”

Questa è la pagina che viene visualizzata subito dopo l'accesso al sistema:



SCHERMATA 5.3

5.2.5 Valutazione “Informazioni iniziali”

Visibilità dello stato del sistema:

La barra di stato indica che ci si trova sulla 'Pagina iniziale'.

Corrispondenza fra il mondo reale e il sistema:

Le informazioni testuali presenti nella schermata centrale sono espresse con parole e frasi semplici quasi colloquiali.

Libertà e controllo da parte degli utenti:

Se l'utente esce (anche per sbaglio) da questa pagina e vuole ritornarci, può cliccare sul menu '?' e poi su 'Pagina iniziale' come spiegato anche nelle informazioni di benvenuto. Questa evita quella situazione di disagio che si prova quando si abbandona una pagina solo per provare a navigare nell'interfaccia e poi non si riesce a ritornarvi.

Consistenza e standard:

La visualizzazione del 'Contenitore principale' rimane sempre coerente alla progettazione del paragrafo precedente (cambiano soltanto le informazioni centrali). L' 'Help' è presentato con una struttura ad albero che è familiare all'utente perché presente nei più diffusi sistemi operativi.

Prevenzione degli errori:

Nelle informazioni di benvenuto è spiegato come accedere di nuovo alle 'Guide veloci' e al 'Manuale d'uso' una volta abbandonata la 'Pagina iniziale' per evitare che l'utente esca da tale pagina e non sappia più come ritornare alle informazioni di 'Help'.

L'intera 'Pagina iniziale' è stata creata con lo scopo di minimizzare gli errori durante il primo utilizzo del sistema attraverso la consultazione delle 'Guide veloci' che descrivono brevemente come effettuare le operazioni più frequenti.

Riconoscere piuttosto che ricordare:

L'utente non deve necessariamente ricordare come riottenere le informazioni di 'Help'. In qualsiasi punto della navigazione all'interno dell'interfaccia, l'utente ottiene tutte le informazioni di aiuto riconoscendo la voce di menu '?' comune a tutte le applicazioni a finestre.

Flessibilità ed efficienza d'uso:

L'utente esperto può tranquillamente ignorare le informazioni iniziali dopo l'accesso e cominciare subito ad operare sui pulsanti del 'Contenitore principale'. Potrebbe essere prevista un opzione (in

questo caso non implementata) reperibile tramite la voce 'Strumenti' della 'Menubar' che consenta all'utente esperto di decidere la schermata iniziale da visualizzare dopo l'accesso.

Design minimalista:

Il testo di benvenuto è breve e ci sono solo le informazioni essenziali per non togliere la visibilità alle informazioni rilevanti (il menu dei pulsanti). Anche il menu ad albero al nodo delle 'Guide veloci' è molto snello per non intimorire l'utente e invogliarlo a leggere.

Guida e documentazione:

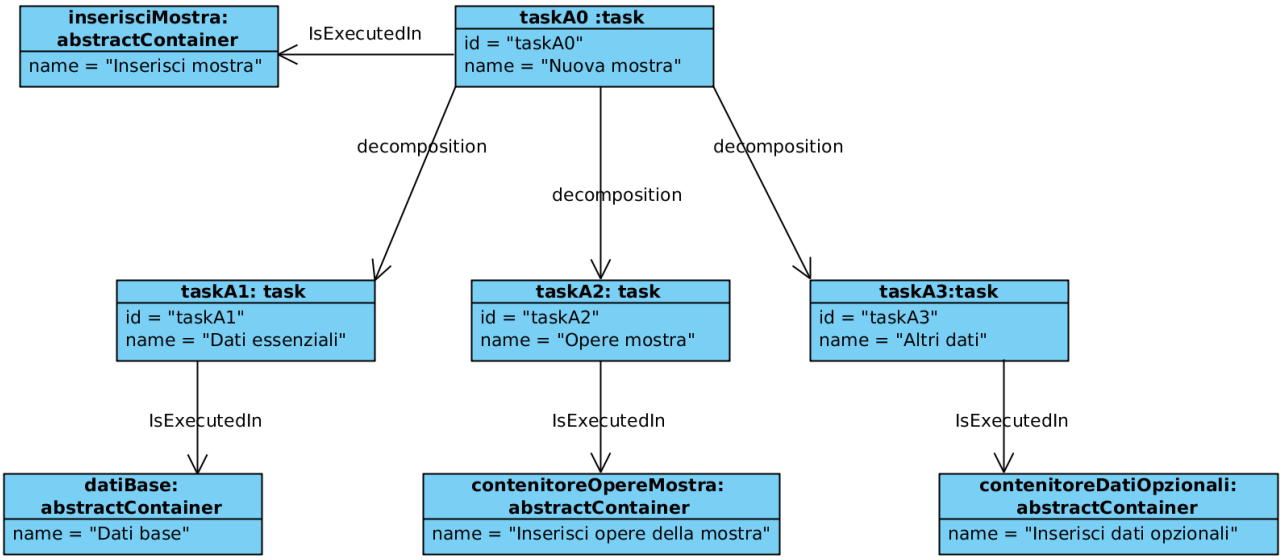
Si deve tener presente che gli utenti spesso evitano di consultare le guide che considerano noiose. Creando le 'Guide veloci' in aggiunta al classico 'Manuale d'uso' si vuole invogliare l'utente a consultarle vista la loro brevità, esse sono focalizzate a questo scopo sui compiti più comuni che può svolgere l'utente e solo quando l'utente ne avrà realmente bisogno consulterà la documentazione più estesa del 'Manuale d'uso'. Tuttavia bisogna anche considerare che l'utente potrebbe non leggere neanche le 'Guide veloci' quindi la progettazione dell'interfaccia è stata comunque effettuata nell'ipotesi che l'utente non le abbia consultate.

Come abbiamo già detto, 'Guide veloci' e 'Manuale d'uso' sono facilmente raggiungibili: si ottengono subito dopo l'accesso al sistema e si riottengono tramite il menu 'Help'.

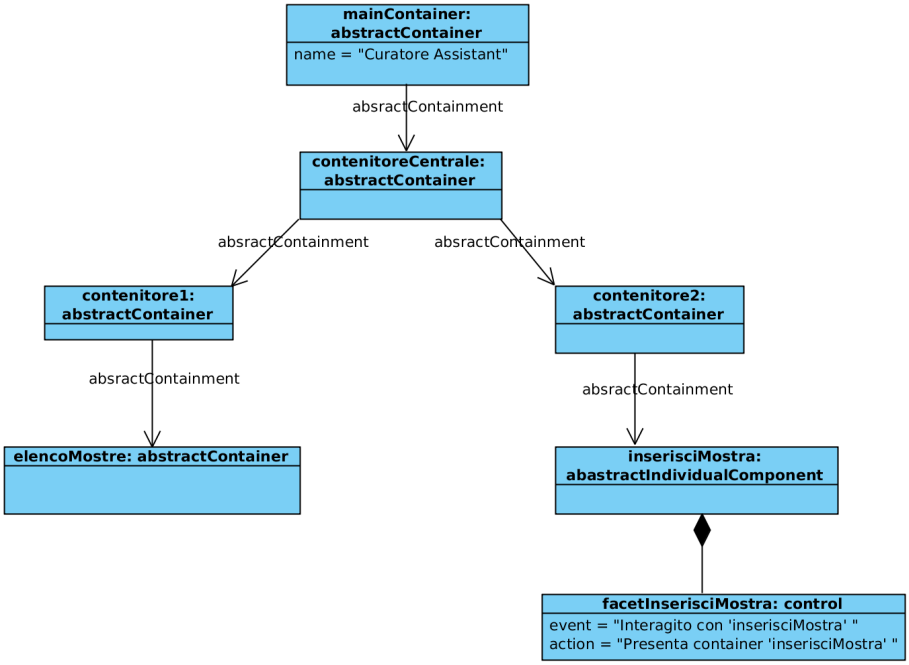
5.3 Realizzazione compito A

5.3.1 Progettazione astratta “Organizzare nuova mostra”

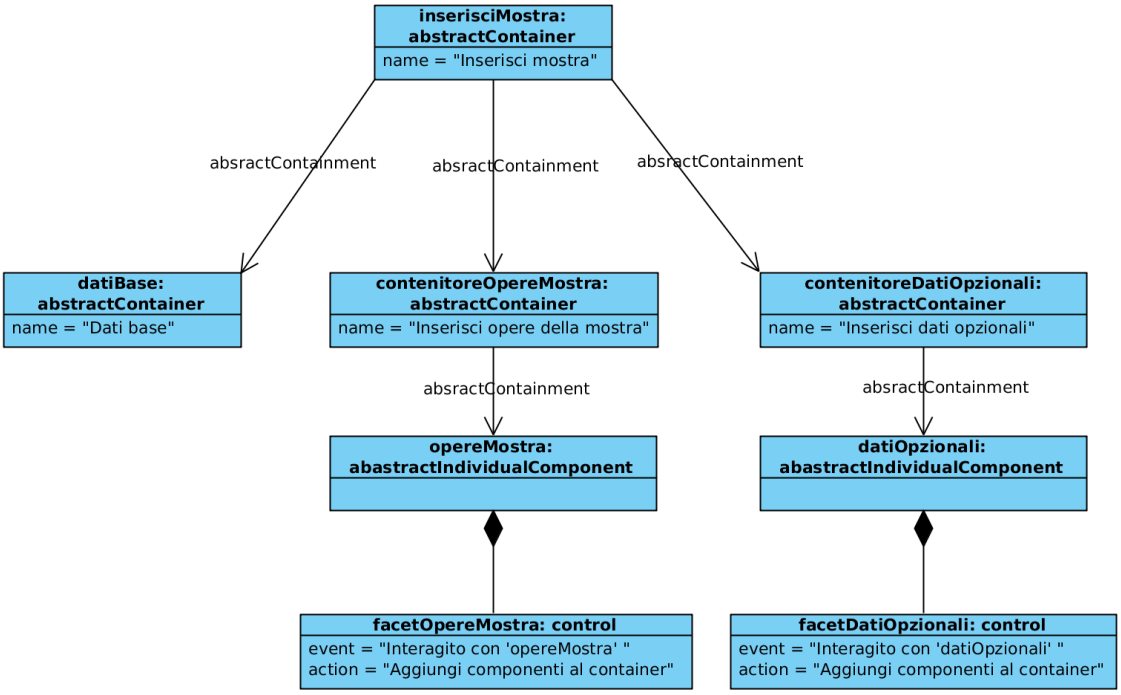
Da questo momento in poi la progettazione astratta dell'interfaccia d'uso sarà ricavata dall'analisi dei compiti. Verrà descritto il relativo diagramma dei compiti del capitolo 3 in notazione UsiXML (Task model) e verranno mostrati quali sono i contenitori astratti relativi ad un particolare sottocompito tramite la relazione 'IsExecutedIn':



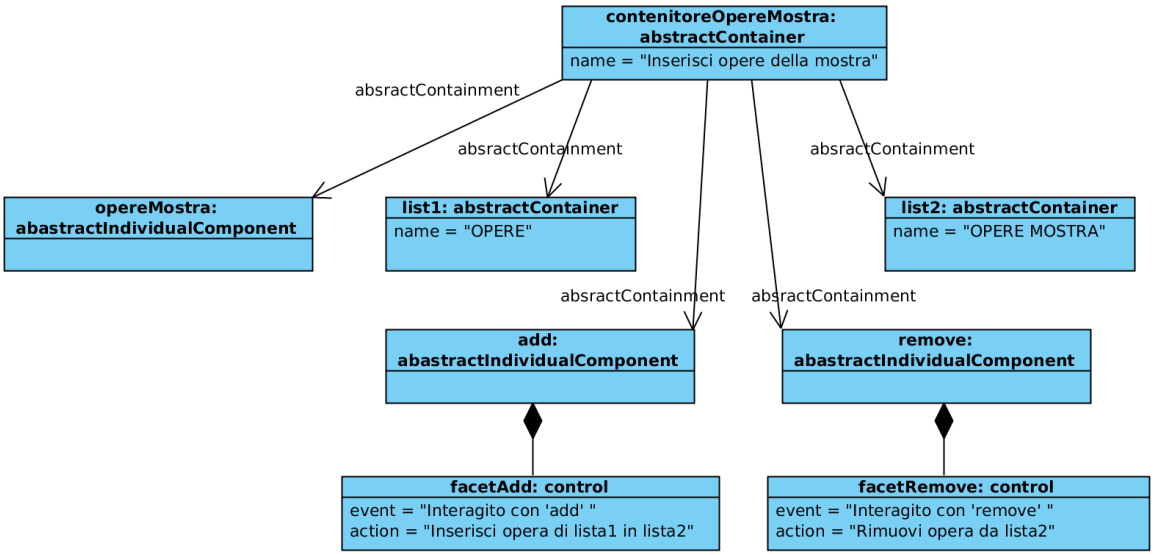
La progettazione della sezione di interfaccia relativa alle 'Mostre', che contiene il riferimento al contenitore astratto 'inserisciMostra', completa lo schema del paragrafo 5.1.1 andando a *riempire* il 'contenitoreCentrale' nel momento in cui si accede a questa parte di interfaccia. In particolare abbiamo il 'contenitore 1' che contiene le informazioni sulle mostre attualmente presenti sul sistema e il 'contenitore 2' che contiene il controllo sulle mostre tra cui 'inserisciMostra' il quale realizza il compito in esame. Vediamone il diagramma:



Vediamo adesso la struttura astratta del *container* 'inserisciMostra':

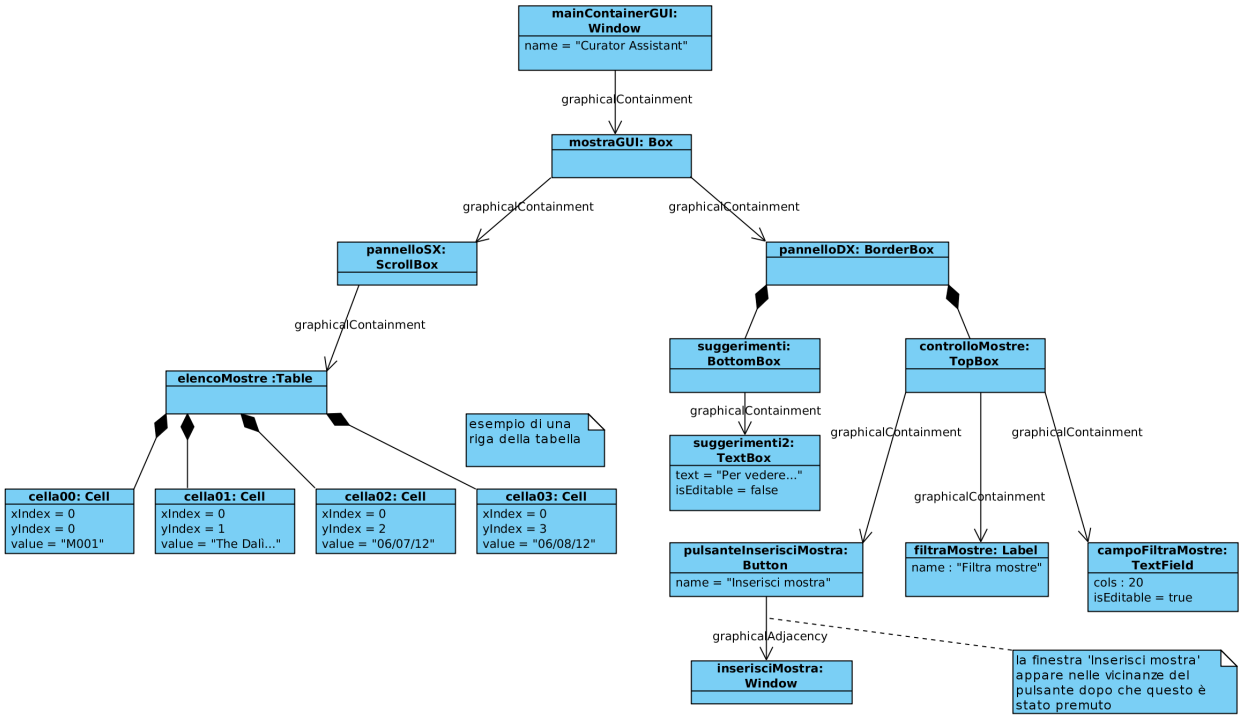


Interagendo con il componente individuale astratto 'opereMostra' si aggiungono nuovi componenti al contenitore che permettono la scelta delle opere associate alla mostra:

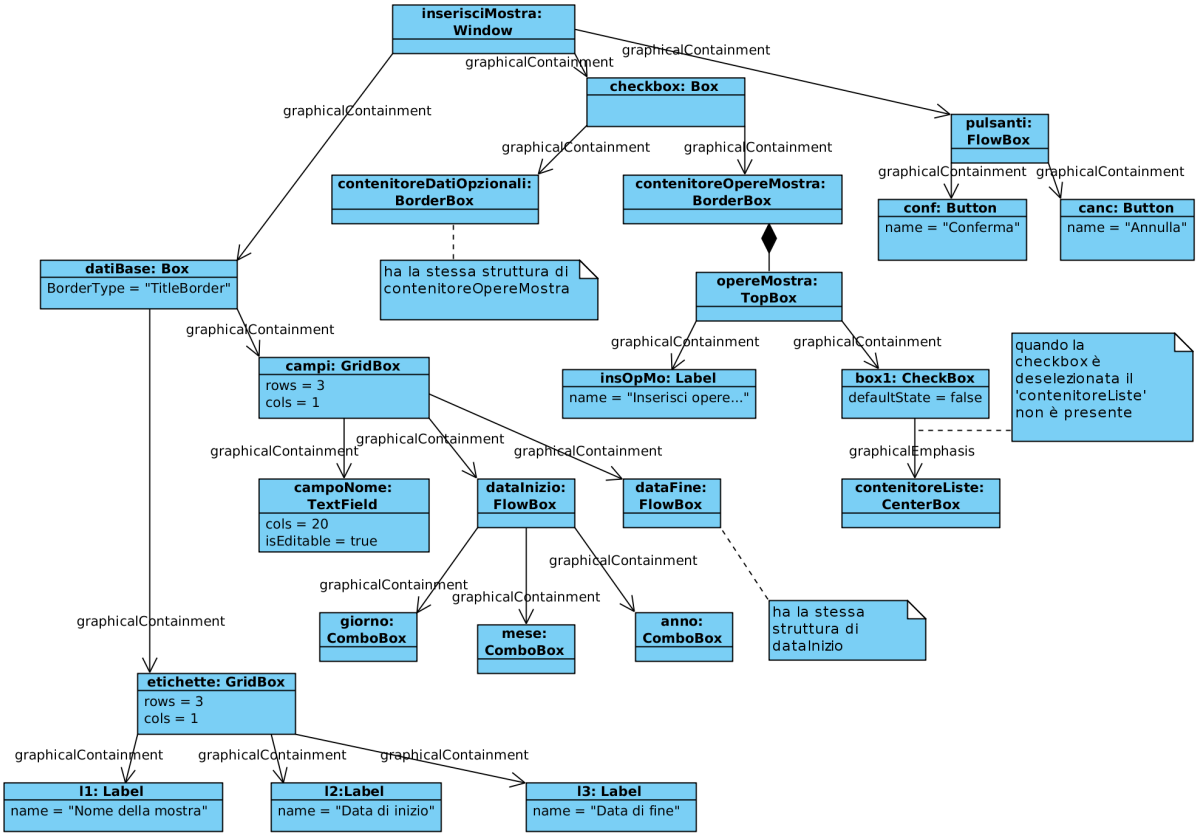


5.3.2 Progettazione concreta "Organizzare nuova mostra"

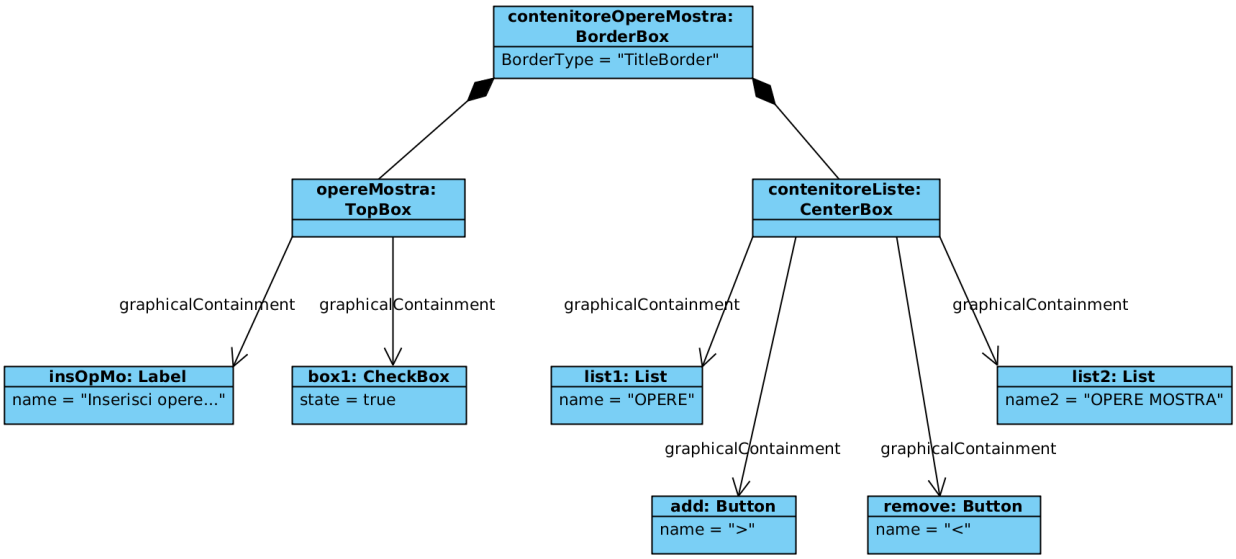
La progettazione concreta oltre ad essere scaturita dall'analisi dei compiti tramite la progettazione astratta è guidata anche dallo "Scenario 1" del capitolo 4.



Vediamo in dettaglio la struttura concreta della finestra 'inserisciMostra':



Se viene selezionata la *checkbox* 'Inserisci opere della mostra', il 'contenitoreOpereMostra' si espande consentendo all'utente l'aggiunta delle opere che parteciperanno alla mostra. Vediamo quali sono i componenti che si aggiungono a tale contenitore durante questa attività:



5.3.3 Implementazione fisica “Organizzare nuova mostra”

Per visualizzare la schermata relativa alle mostre completare il 'Frammento omesso 2' del codice 5.1.3 con il seguente frammento, oltre a quanto già aggiunto nei paragrafi precedenti:

```
final MostraGUI panMostra = new MostraGUI(this);
pulsante2.addActionListener(new AscoltatorePulsanti(this, panMostra,pulsante2));
```

Aggiungere in coda alla classe 'MainContainerGUI' la seguente classe:

```
class AscoltatorePulsanti implements ActionListener{

        private MainContainerGUI fin;
        private Object ob;
        private JButton bot;

        public AscoltatorePulsanti(MainContainerGUI win, Object o, JButton b) {

                fin=win;
                ob=o;
                bot=b;

        }

        public void actionPerformed(ActionEvent e){


                if (e.getActionCommand().compareTo("Mostre")==0) {

                   fin.remove(fin.pannelloSX);
                   fin.remove(fin.pannelloDX);

                    MostraGUI panMostra;
                    panMostra=(MostraGUI)ob;
                    panMostra.ridisegna();

                    fin.add(fin.pannelloSX, BorderLayout.WEST);
                    fin.add(fin.pannelloDX,BorderLayout.EAST);

                    fin.repaint();
                    fin.setVisible(true);


                }
        //---------------------------Frammento omesso 4----------------------------------

        //-------------------------------------------------------------------------------
        }

}
```

Aggiungere al package la classe 'MostraGUI':

```
package curatorAssistantGUI_betaVs;
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.awt.event.*;
import java.io.*;

public class MostraGUI {

        Container panDX;
        Container panSX;
        MainContainerGUI menuPrincipale;

        public MostraGUI(MainContainerGUI fin){

        panDX = new JPanel(); //JPanel è un sottotipo di Container
        menuPrincipale=fin;

        DefaultTableModel mod = new DefaultTableModel();
        mod.addColumn("Identificativo");
        mod.addColumn("Nome");
        mod.addColumn("Data Inizio");
        mod.addColumn("Data Fine");
       mod.setRowCount(100);

       mod.setValueAt("M001", 0, 0);
       mod.setValueAt("La joie de vivre", 0, 1);
       mod.setValueAt("06-Luglio-2012", 0, 2);
       mod.setValueAt("06-Agosto-2012", 0, 3);

       JTable tab = new JTable(mod){
       @Override        public boolean isCellEditable(int row, int col) {
                                    return false;
                                }
                 };

       panSX = new JScrollPane(tab);
       panSX.setPreferredSize(new Dimension(1000,300));
       JPanel panControlloMostre = new JPanel();
       JPanel contenitorePulsante =new JPanel();
       JPanel contenitoreCampoTesto =new JPanel();
       JPanel suggerimenti =new JPanel();

       panDX.setPreferredSize(new Dimension(300,100));

       panControlloMostre.setLayout(new GridLayout(12,1));

       Image i=java.awt.Toolkit.getDefaultToolkit().getImage(
       "/home/macash/Desktop/CollegamentoIUM/img/plus2.gif");
        i=i.getScaledInstance(15,15,Image.SCALE_DEFAULT);
        ImageIcon ic = new ImageIcon(i);

        JButton pulsInsertMostra = new JButton(" Inserisci nuova mostra");
       pulsInsertMostra.setIcon(ic);
       pulsInsertMostra.addActionListener(new MyInsMostraActionListener(tab));

       JLabel lab = new JLabel();
       panControlloMostre.add(lab);
       contenitorePulsante.add(pulsInsertMostra);
       panControlloMostre.add(contenitorePulsante);
       panControlloMostre.add(new JLabel());
       panControlloMostre.add(new JLabel("Filtra Mostre", JLabel.CENTER));
       JTextField t = new JTextField(20);
       contenitoreCampoTesto.add(t);
       panControlloMostre.add(contenitoreCampoTesto);

       JTextPane tp = new JTextPane();
       tp.setFont(new Font(tp.getFont().getName(), tp.getFont().getStyle(), 11 ));
String s1 = new String("Suggerimenti: \n- Per vedere i dati estesi di una mostra\n  fai doppio click ");
String s2 = new String("sulla riga corrispondente.\n  Nei dati estesi trovi nuove funzioni come:");
String s3 = new String("\n  MODIFICA MOSTRA \n  VEDI SCHEDA ILLUSTRATIVA\n\n -Per filtrare le mostre scrivi");
tp.setText(s1+s2+s3+" una parola\n  nel campo \"Filtra mostre\" e premi INVIO.");
       tp.setEditable(false);
       tp.setBackground(panDX.getBackground());
       suggerimenti.add(tp);
       suggerimenti.setBorder(BorderFactory.createLoweredBevelBorder());
```

```
        panDX.add(panControlloMostre, BorderLayout.NORTH);
        panDX.add(suggerimenti, BorderLayout.SOUTH);


        }

        public void ridisegna(){

                menuPrincipale.pannelloSX=panSX;
                menuPrincipale.pannelloDX=panDX;

        }

}
```

Aggiungere in coda alla classe 'MostraGUI' le seguenti classi:

```
class MyInsMostraActionListener implements ActionListener{

        JTable tab;
        JComboBox[] vectorCB = new JComboBox[6];

        public MyInsMostraActionListener(JTable t){tab=t;}

        public void actionPerformed(ActionEvent e){

                JFrame inserisciMostra = new JFrame("Inserisci mostra");

                JPanel etichette = new JPanel(new GridLayout(4,1));
                etichette.add(new JLabel(" Nome della mostra:"+" "+" "+" "+" "+" "+" "));
                etichette.add(new JLabel(" Data di inizio:"));
                etichette.add(new JLabel(" Data di fine:"));

                JPanel campi = new JPanel(new GridLayout(4, 1));
                JPanel text1 = new JPanel();
                JTextField campoNome = new JTextField(20);
                text1.add(campoNome);
                campi.add(text1);

                DefaultComboBoxModel model1 = new DefaultComboBoxModel();
                model1.addElement("01");
                model1.addElement("02");
                model1.addElement("03");
                model1.addElement("04");
                model1.addElement("05");
                model1.addElement("06");
                JComboBox giorno = new JComboBox(model1);

                DefaultComboBoxModel model2 = new DefaultComboBoxModel();
                model2.addElement("Settembre");
                model2.addElement("Ottobre");
                model2.addElement("Novembre");
                model2.addElement("Dicembre");
                JComboBox mese = new JComboBox(model2);

                DefaultComboBoxModel model3 = new DefaultComboBoxModel();
                model3.addElement("2012");
                model3.addElement("2013");
                model3.addElement("2014");
                JComboBox anno = new JComboBox(model3);

                JPanel dataInizio = new JPanel(new FlowLayout());
                dataInizio.add(giorno);
                dataInizio.add(mese);
                dataInizio.add(anno);
                campi.add(dataInizio);

                DefaultComboBoxModel model1_2 = new DefaultComboBoxModel();
                model1_2.addElement("01");
                model1_2.addElement("02");
                model1_2.addElement("03");
                model1_2.addElement("04");
                model1_2.addElement("05");
                model1_2.addElement("06");
                JComboBox giorno2 = new JComboBox(model1_2);
```

```
                DefaultComboBoxModel model2_2 = new DefaultComboBoxModel();
                model2_2.addElement("Settembre");
                model2_2.addElement("Ottobre");
                model2_2.addElement("Novembre");
                model2_2.addElement("Dicembre");
                JComboBox mese2 = new JComboBox(model2_2);

                DefaultComboBoxModel model3_2 = new DefaultComboBoxModel();
                model3_2.addElement("2012");
                model3_2.addElement("2013");
                model3_2.addElement("2014");
                JComboBox anno2 = new JComboBox(model3_2);

                JPanel dataFine = new JPanel(new FlowLayout());
                dataFine.add(giorno2);
                dataFine.add(mese2);
                dataFine.add(anno2);
                campi.add(dataFine);

                Box gruppo = Box.createHorizontalBox();
                gruppo.add(etichette);
                gruppo.add(campi);
                gruppo.setBorder(BorderFactory.createTitledBorder("DATI BASE"));

                JPanel datiBase = new JPanel(new FlowLayout(FlowLayout.LEFT));
                datiBase.add(gruppo);

                JPanel contenitoreOpereMostra = new JPanel(new BorderLayout());
                JPanel opereMostra = new JPanel(new FlowLayout(FlowLayout.LEFT));
                JLabel insOp = new JLabel("Inserisci opere della mostra: "+" ");
                JCheckBox box1 = new JCheckBox();
                opereMostra.add(insOp);
                opereMostra.add(box1);
                contenitoreOpereMostra.add(opereMostra, BorderLayout.NORTH);

                box1.addItemListener(new MyItemListener(contenitoreOpereMostra,inserisciMostra,box1));

                JPanel contenitoreDatiOpzionali = new JPanel(new BorderLayout());
                JPanel datiOpzionali = new JPanel(new FlowLayout(FlowLayout.LEFT));
                JLabel datiOpt = new JLabel("Inserisci dati opzionali: "+" ");
                JCheckBox box2 = new JCheckBox();
                datiOpzionali.add(datiOpt);
                datiOpzionali.add(box2);
                contenitoreDatiOpzionali.add(datiOpzionali, BorderLayout.NORTH);

                JPanel checkbox = new JPanel(new FlowLayout(FlowLayout.LEFT));
                checkbox.add(contenitoreOpereMostra, BorderLayout.NORTH);
                checkbox.add(contenitoreDatiOpzionali, BorderLayout.SOUTH);

                JPanel pulsanti = new JPanel(new FlowLayout());
                JButton canc = new JButton("Annulla");
                JButton conf = new JButton("Conferma");

                vectorCB[0]=giorno;
                vectorCB[1]=mese;
                vectorCB[2]=anno;
                vectorCB[3]=giorno2;
                vectorCB[4]=mese2;
                vectorCB[5]=anno2;
                conf.addActionListener(new MyConfActionListener(campoNome,tab, vectorCB, inserisciMostra));

                pulsanti.add(canc);
                pulsanti.add(conf);

                conf.setEnabled(false);
                campoNome.addKeyListener(new MyKeyListener(campoNome,conf));

                inserisciMostra.add(datiBase,BorderLayout.NORTH);
                inserisciMostra.add(checkbox,BorderLayout.CENTER);
                inserisciMostra.add(pulsanti,BorderLayout.SOUTH);

                inserisciMostra.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
                inserisciMostra.pack();
                inserisciMostra.setSize(420,400);
                inserisciMostra.setVisible(true);

        }

}
```

```
class MyKeyListener implements KeyListener {

        JTextField tx;
        JButton bot;

        MyKeyListener(JTextField t, JButton b){ tx=t; bot=b;}


                        public void keyPressed(KeyEvent e){}
                        public void keyTyped(KeyEvent e){}
                        public void keyReleased(KeyEvent e){

                                if(!tx.getText().isEmpty()) { bot.setEnabled(true);}
                                else bot.setEnabled(false);

                        }
}
```

```
class MyItemListener implements ItemListener {

                Boolean flag=false;
                JPanel contOpMostra;
                JFrame fin;
                JCheckBox box;
                JTextPane opereTp = new JTextPane();
                JTextPane opereMostraTp = new JTextPane();
                JPanel contenitoreListe = new JPanel();
                DefaultListModel listmod = new DefaultListModel();
                DefaultListModel listmod2 = new DefaultListModel();
                JList list = new JList(listmod);
                JList list2 = new JList(listmod2);
                JButton add = new JButton(">");
                JButton remove = new JButton("<");
                JPanel contBut = new JPanel(new GridLayout(3,1));
                JLabel l = new JLabel();
                Object elemSel;
                JPanel contList1 = new JPanel();
                JPanel contList2 = new JPanel();

                public MyItemListener(JPanel p, JFrame f, JCheckBox b){contOpMostra=p; fin=f; box=b;}

                public void itemStateChanged(ItemEvent e){

                        if(flag==null){}
                        else {
                                if(flag==false) flag=true;
                                else flag=false;

                                if(flag==true){

                                        fin.setSize(650,800);

                                        contOpMostra.setBorder(BorderFactory.createTitledBorder(""));

                                        listmod.addElement("\"La persistenza della memoria\"");
                                        list.setPreferredSize(new Dimension(250,300));
                                        list2.setPreferredSize(new Dimension(250,300));

                                        contBut.add(add);
                                        contBut.add(l);
                                        contBut.add(remove);
                                        remove.setEnabled(false);



                                        contList1.add(list);

                                        contList2.add(list2);


        contList1.setBorder(BorderFactory.createTitledBorder(BorderFactory.createEmptyBorder(),"OPERE:"));

        contList2.setBorder(BorderFactory.createTitledBorder(BorderFactory.createEmptyBorder(),
       "OPERE MOSTRA:"));
```

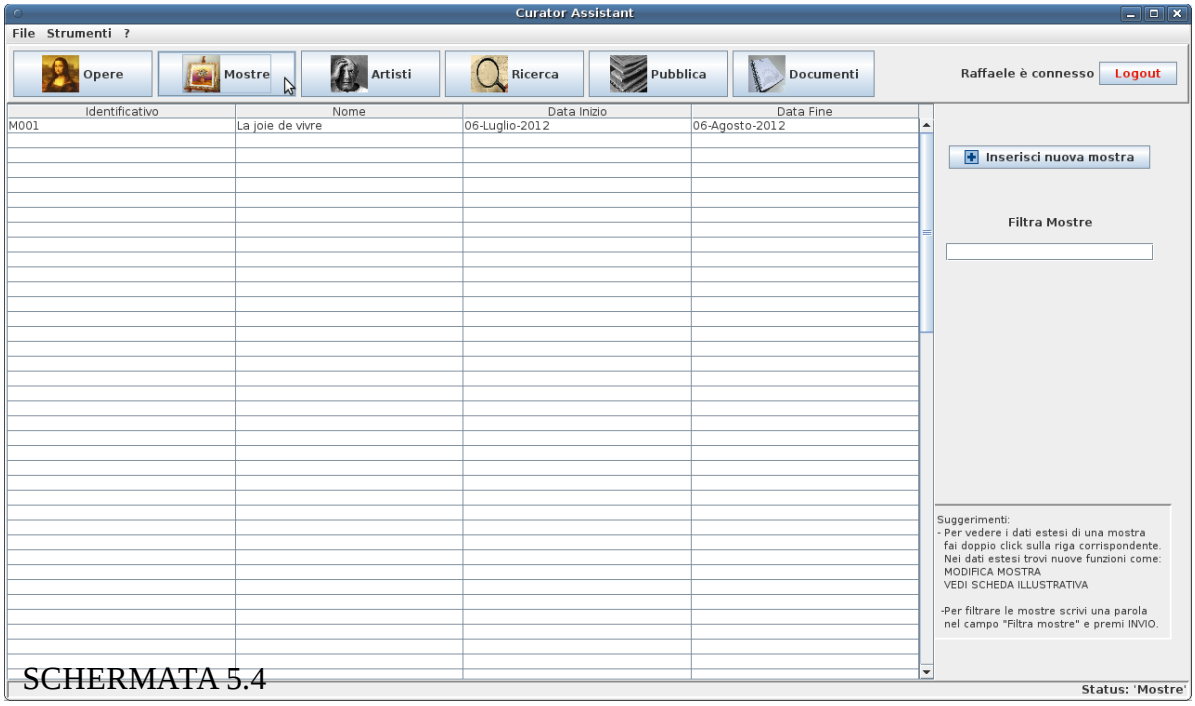
```
                contenitoreListe.add(contList1, BorderLayout.WEST);
                contenitoreListe.add(contBut, BorderLayout.CENTER);
                contenitoreListe.add(contList2, BorderLayout.EAST);
                contOpMostra.add(contenitoreListe, BorderLayout.CENTER);

                list.addMouseListener(new MouseAdapter(){

                        public void mousePressed(MouseEvent e){

                        elemSel =selezionaElemento((JList)e.getSource(),e.getPoint());

                        }

                });

                list2.addMouseListener(new MouseAdapter(){

                        public void mousePressed(MouseEvent e){

                        elemSel =selezionaElemento((JList)e.getSource(),e.getPoint());

                        }

                });

                add.addActionListener(new ActionListener(){

                        public void actionPerformed(ActionEvent e){

                                if (elemSel!=null){

                                        listmod2.addElement(elemSel);
                                        remove.setEnabled(true);
                                        listmod.removeElement(elemSel);
                                        add.setEnabled(false);
                                        elemSel=null;

                                }
                                /*else { JOptionPane.showMessageDialog(add,
                                  "Seleziona un'opera dalla lista di sinistra");}*/

                        };

                });

                remove.addActionListener(new ActionListener(){

                        public void actionPerformed(ActionEvent e){

                                if (elemSel!=null){

                                        listmod.addElement(elemSel);
                                        add.setEnabled(true);
                                        listmod2.removeElement(elemSel);
                                        remove.setEnabled(false);
                                        elemSel=null;

                                }
                        };

                });

        }
        else {
                boolean svuota=true;

                if(!listmod2.isEmpty()){
                        flag=null;
                        int risp = JOptionPane.showConfirmDialog(contOpMostra,
                        "La mostra tornerà senza opere associate",
                        "Attenzione",JOptionPane.OK_CANCEL_OPTION);
                        if(risp==JOptionPane.CANCEL_OPTION){svuota=false; flag=true;}
                        if(risp==JOptionPane.OK_OPTION){
                           box.setSelected(false);
                           flag=false;}
                }

                if(svuota==true){
                        contOpMostra.remove(contenitoreListe);
```

```
                                                elemSel=null;
                                                if(!listmod.isEmpty()){listmod.removeElementAt(0);}
                                                if(!listmod2.isEmpty()){listmod2.removeElementAt(0);
                                                        add.setEnabled(true);
                                                        remove.setEnabled(false);

                                                }

                                                contOpMostra.setBorder(BorderFactory.createEmptyBorder());
                                                fin.setSize(420,400);

                                        }
                                }
                        }
                }

                        public static Object selezionaElemento(JList l, Point clickPoint){

                                int index= l.locationToIndex(clickPoint);
                                if (index >=0) {  //se non lo è, NO elemento lista
                                        l.setSelectedIndex(index);
                                        Object selezionato= l.getModel().getElementAt(index);
                                        return selezionato;
                                }
                                else return null;
                        }
}
```

```
class MyConfActionListener implements ActionListener {

        JTextField tx;
        JTable tabella;
        JComboBox[] vectCB;
        JFrame fin;

        public MyConfActionListener(JTextField t, JTable tab, JComboBox[] v, JFrame f){
              tx=t; tabella=tab; vectCB=v; fin=f;}

        public void actionPerformed(ActionEvent e){

                tabella.getModel().setValueAt("M002", 1, 0);

                tabella.getModel().setValueAt(tx.getText(), 1, 1);

                int index=vectCB[0].getSelectedIndex();
                int index1=vectCB[1].getSelectedIndex();
                int index2=vectCB[2].getSelectedIndex();
                String s = vectCB[0].getModel().getElementAt(index).toString();
                String s1 = vectCB[1].getModel().getElementAt(index1).toString();
                String s2 = vectCB[2].getModel().getElementAt(index2).toString();
                tabella.getModel().setValueAt(s+"-"+s1+"-"+s2,1,2);

                int index3=vectCB[3].getSelectedIndex();
                int index4=vectCB[4].getSelectedIndex();
                int index5=vectCB[5].getSelectedIndex();
                String s3 = vectCB[3].getModel().getElementAt(index3).toString();
                String s4 = vectCB[4].getModel().getElementAt(index4).toString();
                String s5 = vectCB[5].getModel().getElementAt(index5).toString();
                tabella.getModel().setValueAt(s3+"-"+s4+"-"+s5,1,3);

                fin.dispose();
                JOptionPane.showMessageDialog(fin, "Inserita una nuova mostra");

        }

}
```

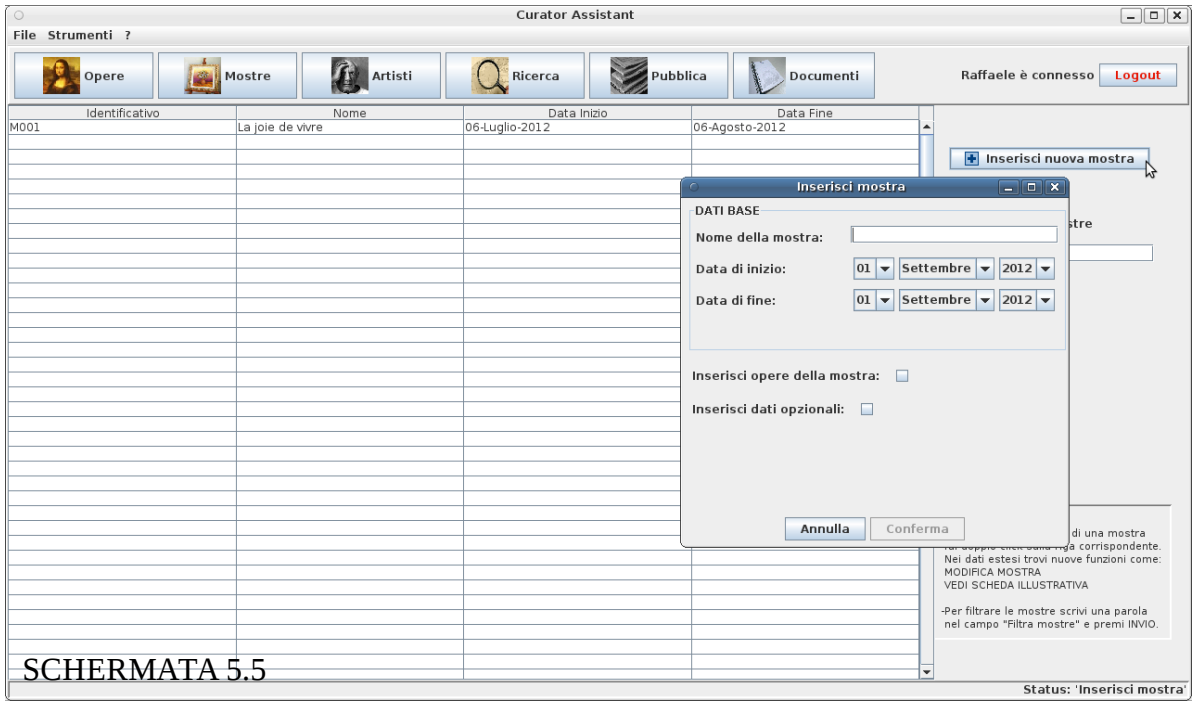
5.3.4 GUI “Organizzare nuova mostra”

Dalla 'Pagina iniziale' Raffaele clicca su 'Mostre', visualizza la schermata seguente in cui in primo piano ci sono le mostre già presenti nel sistema.



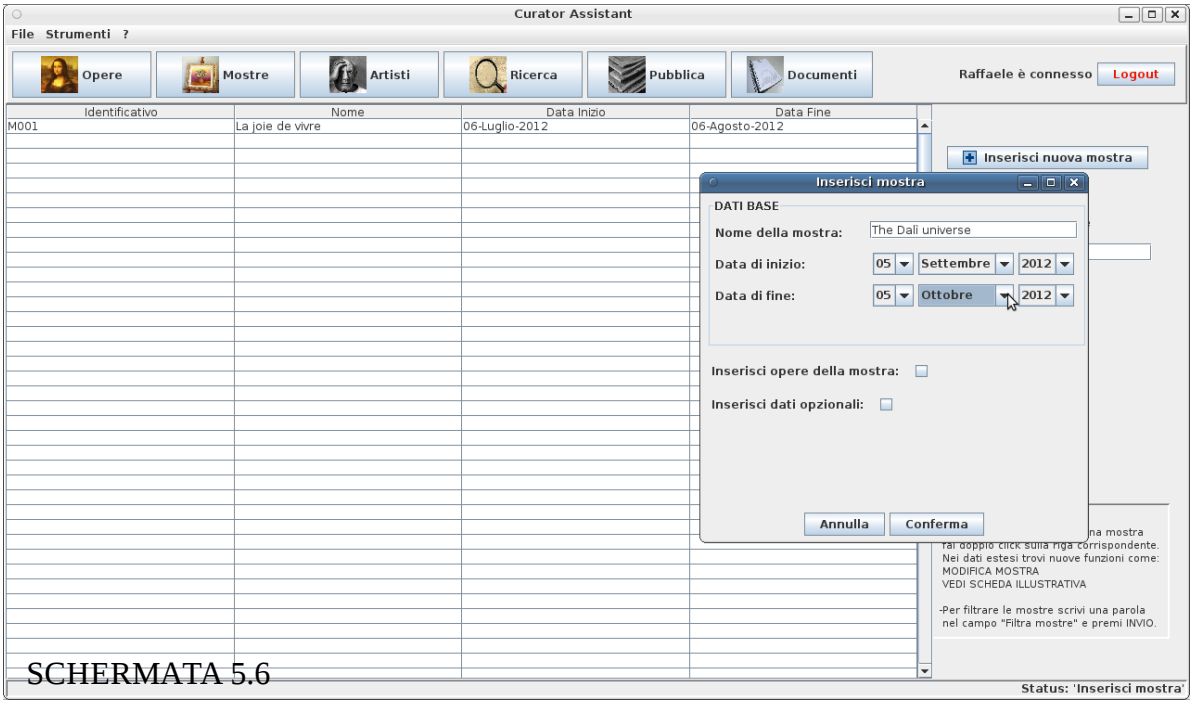
SCHERMATA 5.4

Raffaele clicca su 'Inserisci nuova mostra' e la relativa finestra viene visualizzata in prossimità del pulsante:



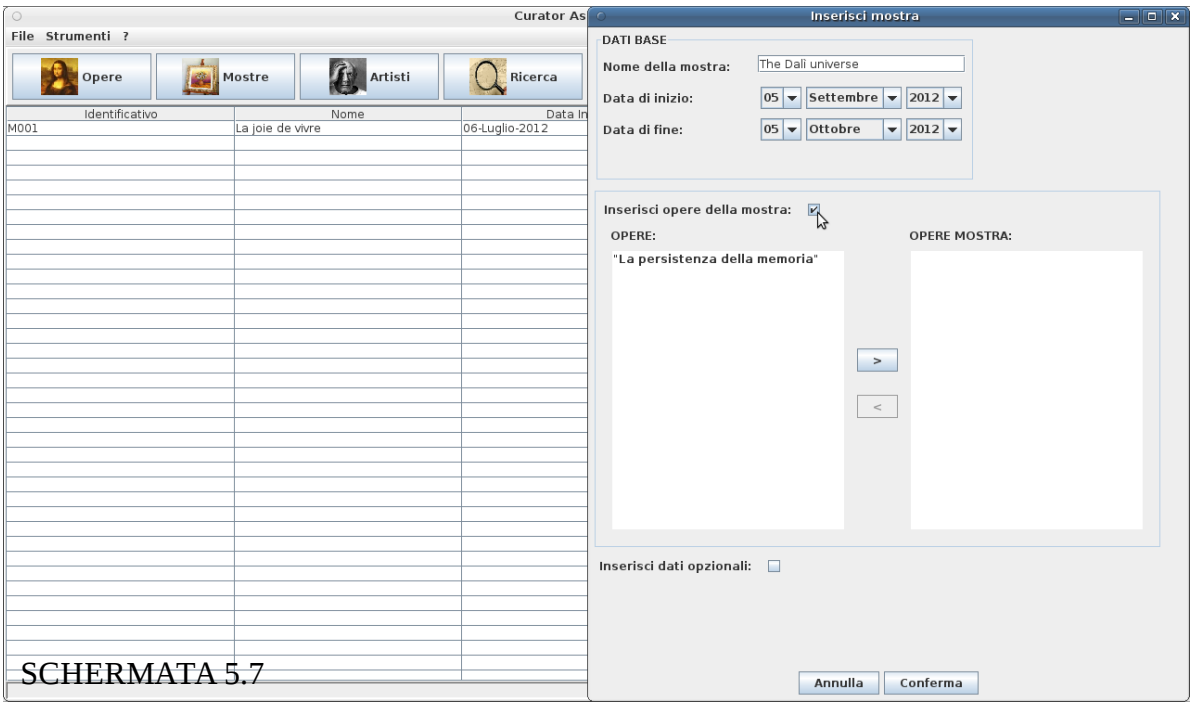
SCHERMATA 5.5

Quando Raffaele inserisce correttamente i 'Dati base' di una nuova mostra, il pulsante 'Conferma' si abilita indicando che tali dati sono sufficienti all'aggiunta di una mostra nel sistema:



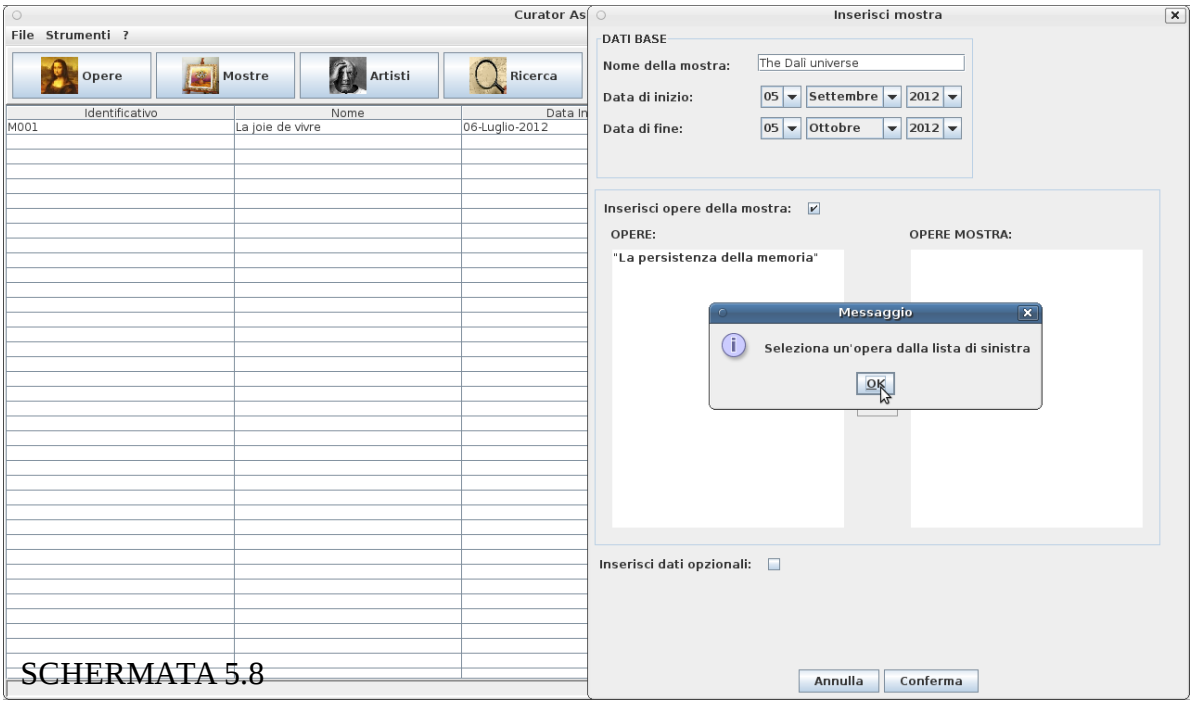
SCHERMATA 5.6

Raffaele decide di inserire le opere che parteciperanno alla mostra e spunta la *checkbox* 'Inserisci opere della mostra', la finestra 'Inserisci mostra' si espande:



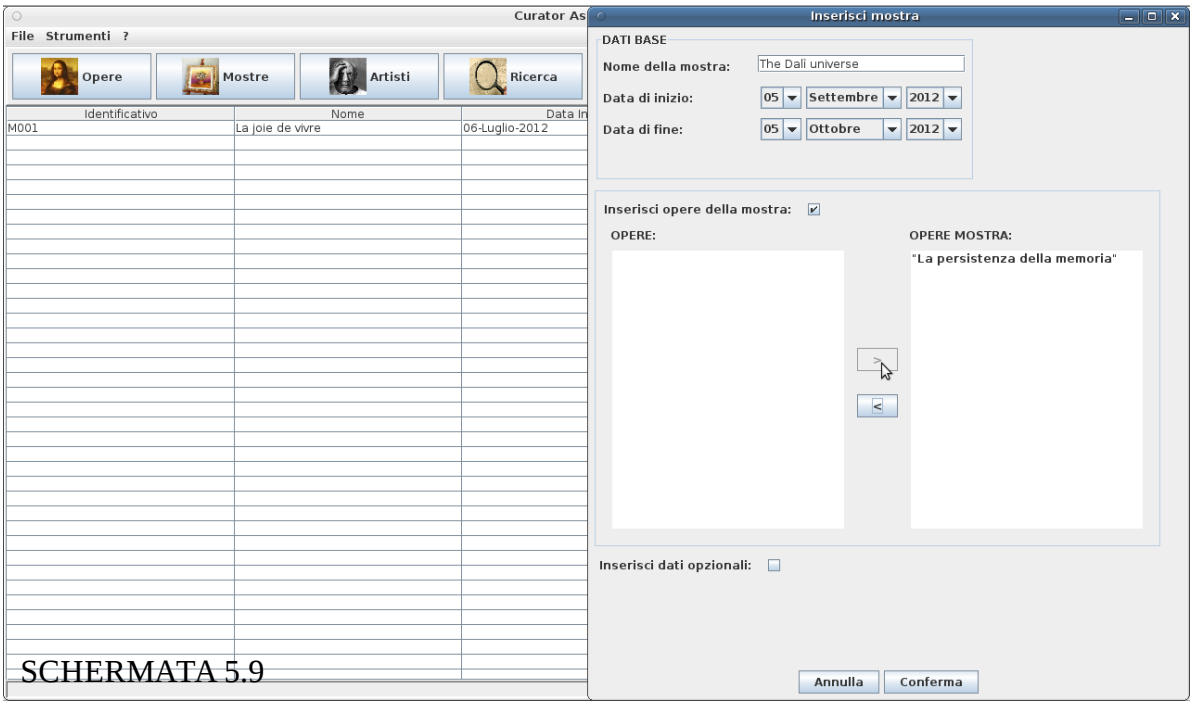
SCHERMATA 5.7

Tramite il pulsante '>' si sposta un'opera dall'elenco di sinistra (elenco di tutte le opere disponibili nel sistema) all'elenco di destra (elenco delle opere associate alla mostra). Raffaele clicca il pulsante '>' senza aver selezionato un'opera dall'elenco di sinistra e viene visualizzata la seguente 'Dialog':



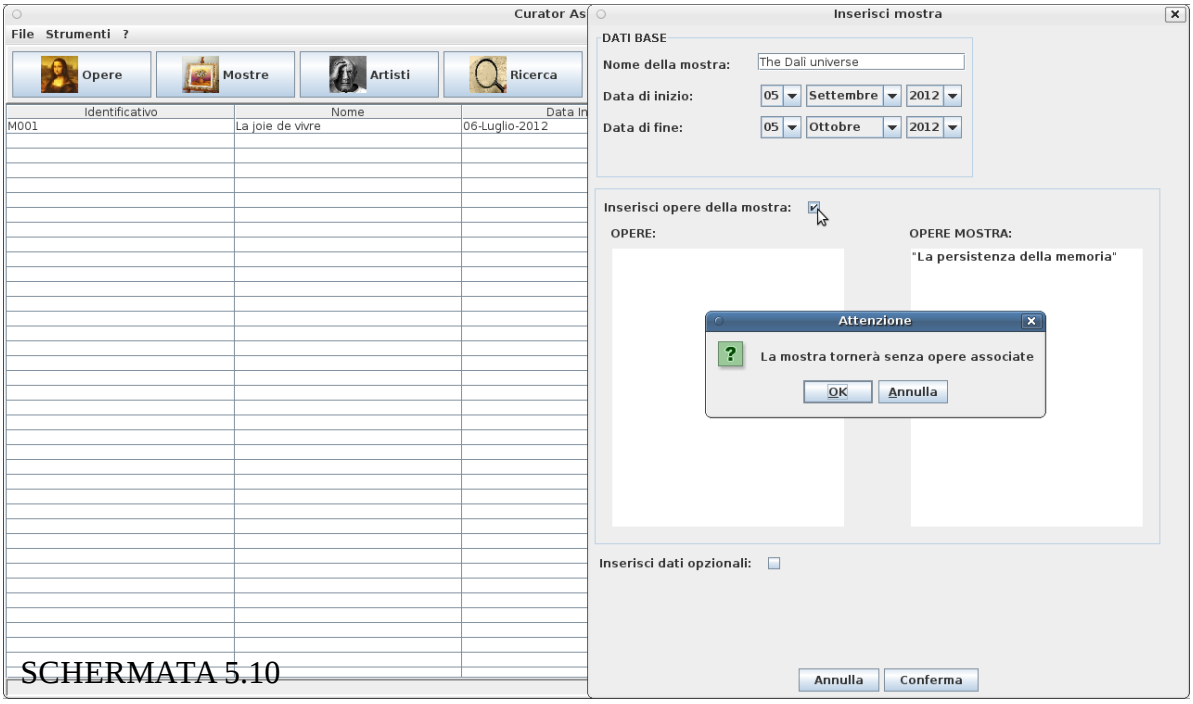
SCHERMATA 5.8

Ripetendo l'operazione con la selezione di un opera, l'opera in questione è spostata nell'elenco di destra e sarà associata alla mostra in creazione:



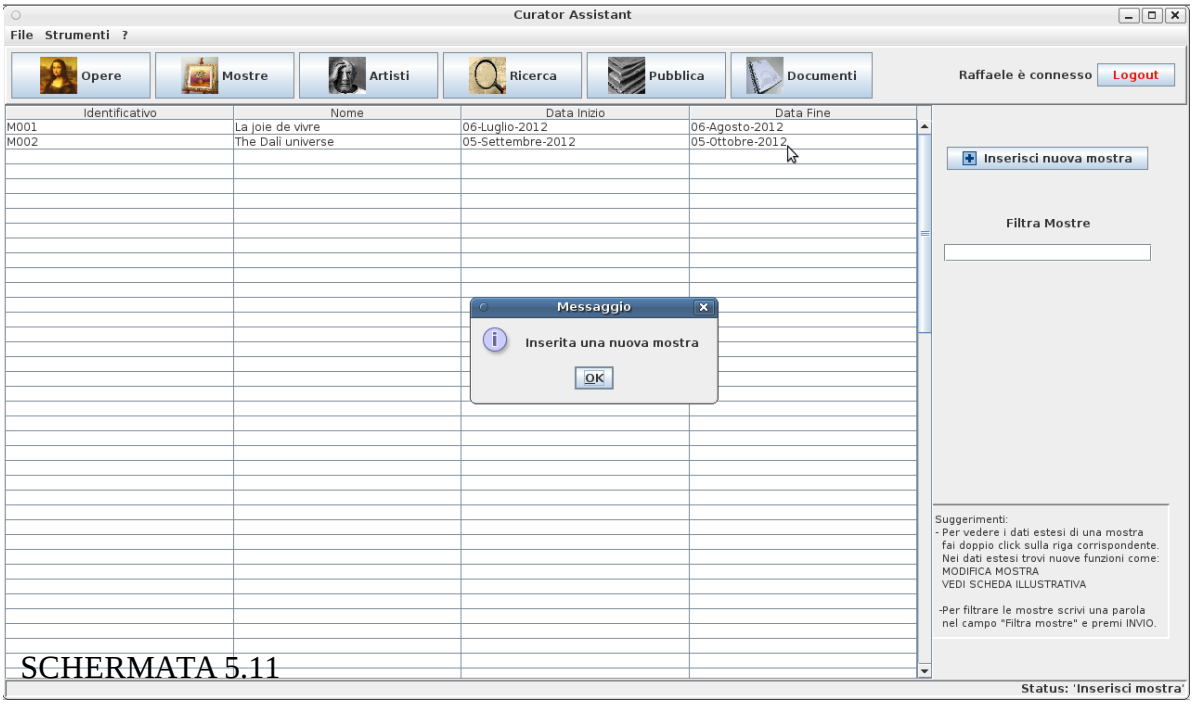
SCHERMATA 5.9

Raffaele deseleziona erroneamente la *checkbox* 'Inserisci opere della mostra' e il sistema lo avvisa che così facendo la mostra perderà le opere associate :



SCHERMATA 5.10

Raffaele annulla e clicca su 'Conferma', il sistema informa Raffaele che una nuova mostra è stata inserita e questa compare nell'elenco delle mostre presenti nel sistema:



SCHERMATA 5.11

5.3.5 Valutazione “Organizzare nuova mostra”

Visibilità dello stato del sistema:

Oltre alla barra di stato che modifica il suo stato a seconda delle azioni dell'utente, il sistema informa sempre l'utente su cosa sta accadendo. Ad esempio nella 'Schermata 5.8' il sistema fa presente all'utente che si trova in uno stato in cui aspetta la selezione di un'opera; oppure nella 'Schermata 5.10' informa l'utente che deselezionando la *checkbox* si perderanno le associazioni tra le opere scelte e la mostra che si sta creando. Infine nella 'Schermata 5.11' si fornisce un *feedback* sull'avvenuto inserimento di una nuova mostra.

Corrispondenza fra il mondo reale e il sistema:

La progettazione di questa parte di interfaccia è focalizzata sul compito di “Organizzare una nuova mostra” ed è quindi ispirata dai passi che Raffaele seguirebbe nella vita reale anche senza l'uso del sistema (decidere i dati essenziali di una nuova mostra corrisponde a compilarne i dati base, decidere le opere che parteciperanno alla mostra corrisponde a selezionare le opere interessate e spostarle nell'elenco delle opere associate alla mostra ecc.). Le informazioni sono presentate a Raffaele secondo un ordine che dovrebbe apparirgli logico e naturale.

Libertà e controllo da parte degli utenti:

Quando l'utente seleziona la *checkbox* 'Inserisci opere della mostra' la finestra si espande e se l'utente deseleziona la *checkbox* senza scegliere alcuna opera per la mostra, la finestra ritorna alle dimensione precedenti. L'utente ha così pieno controllo sulla finestra 'Inserisci mostra', se la esplora modificandone lo stato può tornare allo stato precedente. Un'altra forma di libertà e controllo e l'ordine di inserimento dati, l'utente può inserire prima i dati base (anch'essi in qualsiasi ordine) e poi le opere della mostra o viceversa.

Consistenza e standard:

La sezione della finestra 'Inserisci mostra' dedicata all'inserimento delle opere della mostra con i due riquadri bianchi che possono contenere delle liste e i pulsanti '>' e '<', segue una struttura comunemente adottata nei sistemi operativi e in molte applicazioni a finestre che potrebbe essere già familiare all'utente.

Prevenzione degli errori:

Al fine di evitare che l'utente inserisca dati non validi per la mostra, confermi e poi venga informato che i dati non sono validi e sono da reinserire, il sistema previene l'insorgere di questo problema disabilitando il tasto 'Conferma' finché non sono stati inseriti dati corretti e l'inserimento delle date è tramite *checkbox* per evitare che l'utente scriva le date in formato scorretto (Schermata 5.5). Inoltre il pulsante '<' nella 'Schermata 5.7' è inizialmente disabilitato perché non si può spostare un elemento dall'elenco di destra verso quello di sinistra quando il primo elenco è vuoto quindi con la disabilitazione del pulsante si evita un tentativo di dialogo inconsistente. Infine la *dialog* della 'Schermata 5.10' è stata aggiunta appositamente per evitare che l'utente chiudi l'espansione 'Inserisci opere della mostra' solo per compilare il resto della finestra e poi riselezioni 'Inserisci opere della mostra' o prema 'Conferma' credendo che la mostra abbia ancora le opere associate in precedenza.

Riconoscere piuttosto che ricordare:

L'utente durante l'inserimento di una mostra esegue tre attività, inserire dati base, inserire opere della mostra, inserire dati opzionali; solo la prima attività è obbligatoria ma le tre opzioni sono tutte accessibili dalla stessa finestra 'Inserisci mostra'. In questo modo l'utente non deve ricordare come accedere alle tre opzioni ma le riconosce durante l'inserimento di una mostra e sceglie quale svolgere.

Flessibilità ed efficienza d'uso:

Questa parte di interfaccia è stata progettata pensando all'utente Raffaele, inesperto. Quando si apre

la finestra 'Inserisci mostra' essa è inizialmente di dimensioni ridotte con poche informazioni da inserire (i soli dati base). Questa scelta ha lo scopo di non intimorire l'utente novizio che con pochi passi ottiene il risultato di inserire una mostra. Tuttavia la finestra contiene anche informazioni più complicate che sono all'inizio nascoste. Selezionando 'Inserisci opere della mostra' e 'Inserisci dati opzionali' la finestra si espande aumentando la sua complessità e si adatta così in modo flessibile ad un utilizzo più esperto.

Design minimalista:

La scelta di espandere e comprimere la finestra 'Inserisci mostra' a seconda di ciò che si vuole inserire rispetta i requisiti di un dialogo minimale che in un dato istante mostra all'utente solo le informazioni per esso rilevanti. Tuttavia quando il numero di informazioni nella finestra aumenta, come nell'esempio di 'Schermata 5.7', la lettura e la comprensione da parte dell'utente è facilitata dall'utilizzo da alcune *leggi della Gestalt*. Ad esempio i campi semanticamente correlati sono vicini fra loro, i 'Dati base' sono raggruppati in alto mentre le informazioni sulle opere della mostra sono raggruppate al centro. Ciò aiuta la decomposizione della finestra nelle parti che la compongono perché per la legge della vicinanza *"gli elementi del campo visivo che sono fra loro più vicini tendono a essere raccolti in unità"*. La correlazione fra le informazioni vicine è ulteriormente sottolineata dalle linee che incorniciano i diversi gruppi per sfruttare la legge della chiusura la quale afferma che *"le linee delimitanti una superficie chiusa si percepiscono come unità"*.

Aiutare gli utenti a riconoscere gli errori, diagnosticarli e correggerli:

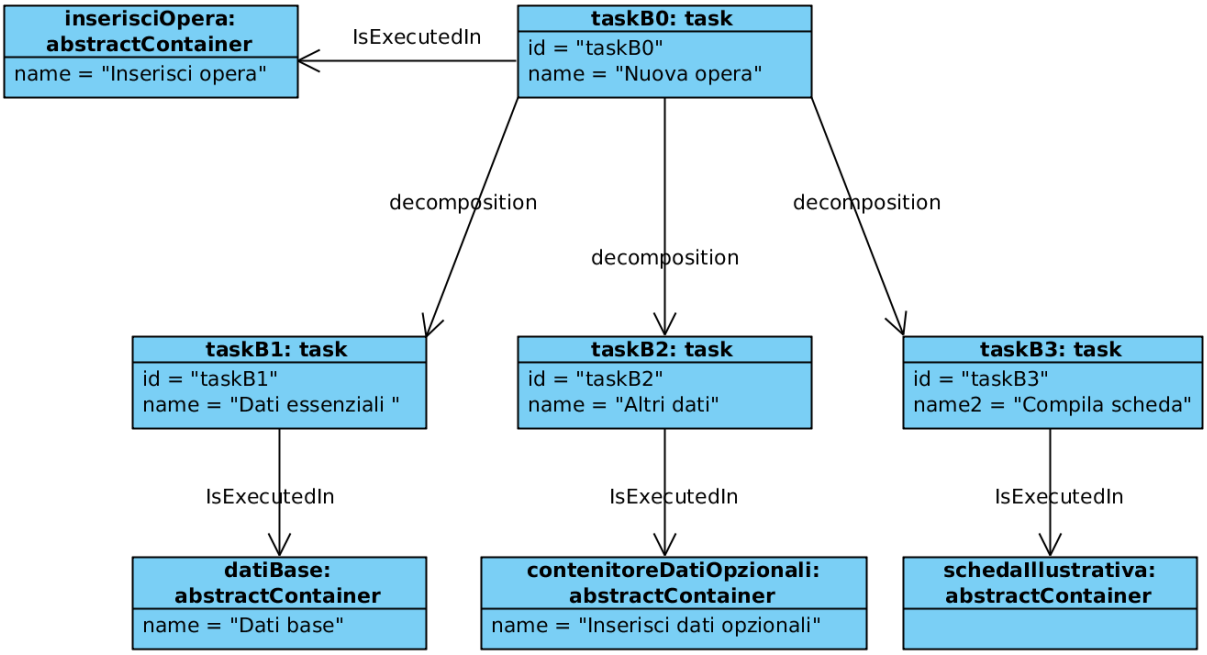
I messaggi di errori nelle *dialog* di 'Schermata 5.8' e 'Schermata 5.10' sono espressi in linguaggio semplice tanto da non sembrare degli errori. Essi non colpevolizzano l'utente ma gli suggeriscono una soluzione, nella 'Schermata 5.8' viene suggerito all'utente di selezionare un'opera dall'elenco di sinistra prima di premere il pulsante '>' mentre nella 'Schermata 5.10' l'utente viene informato che la sua scelta porta all'eliminazione delle associazioni tra opere e mostra, scelta che può correggere premendo 'Annulla'.

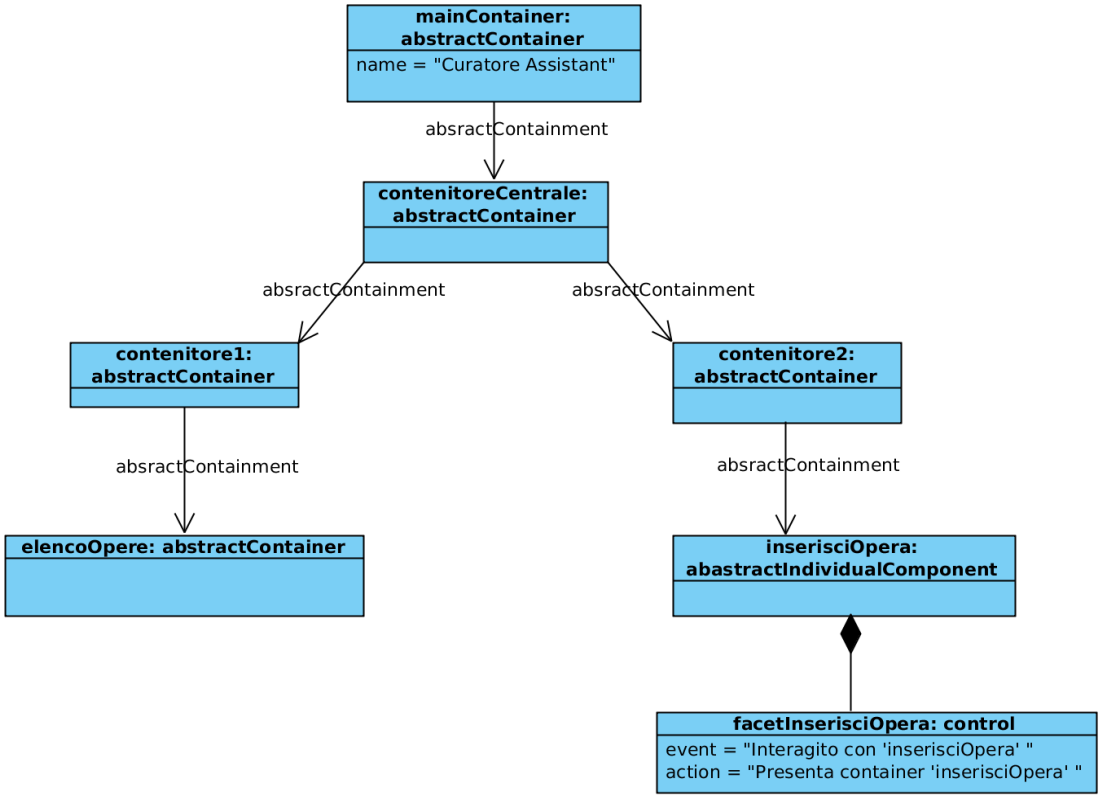
Guida e documentazione:

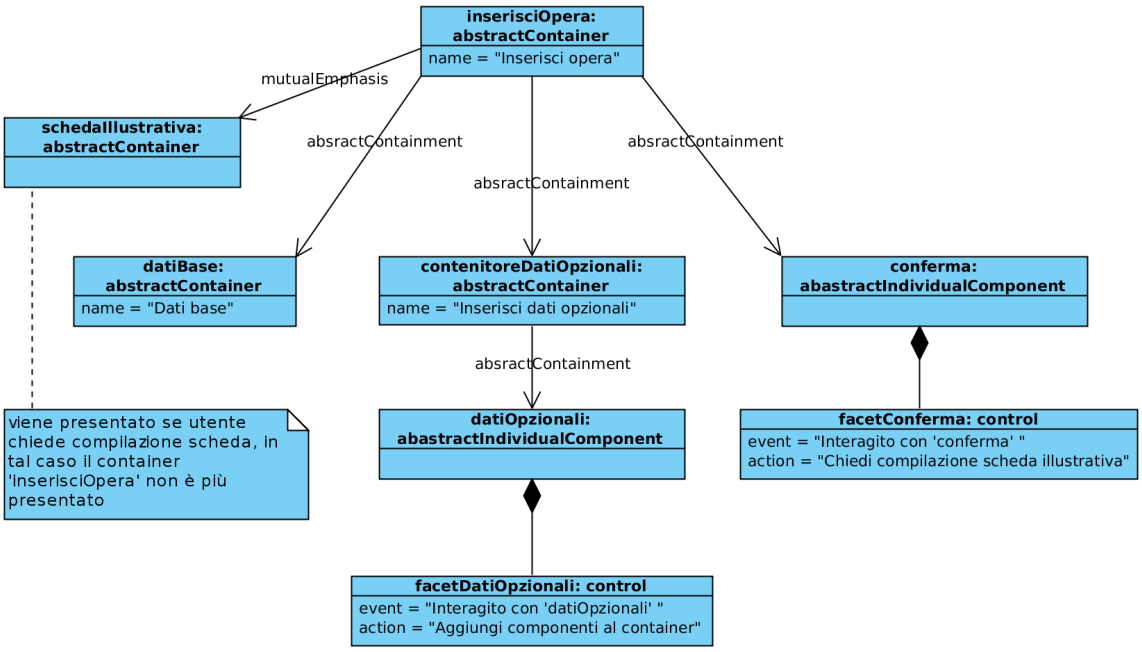
Quando si accede alla sezione 'Mostre' (Schermata 5.4) il menu 'Help' è sempre accessibile tramite la *menubar*, inoltre vengono anche mostrati dei suggerimenti (in basso a destra) che servono all'utente inesperto (da modello Raffaele) per ottenere più facilmente quello di cui ha bisogno.

5.4 Realizzazione compito B

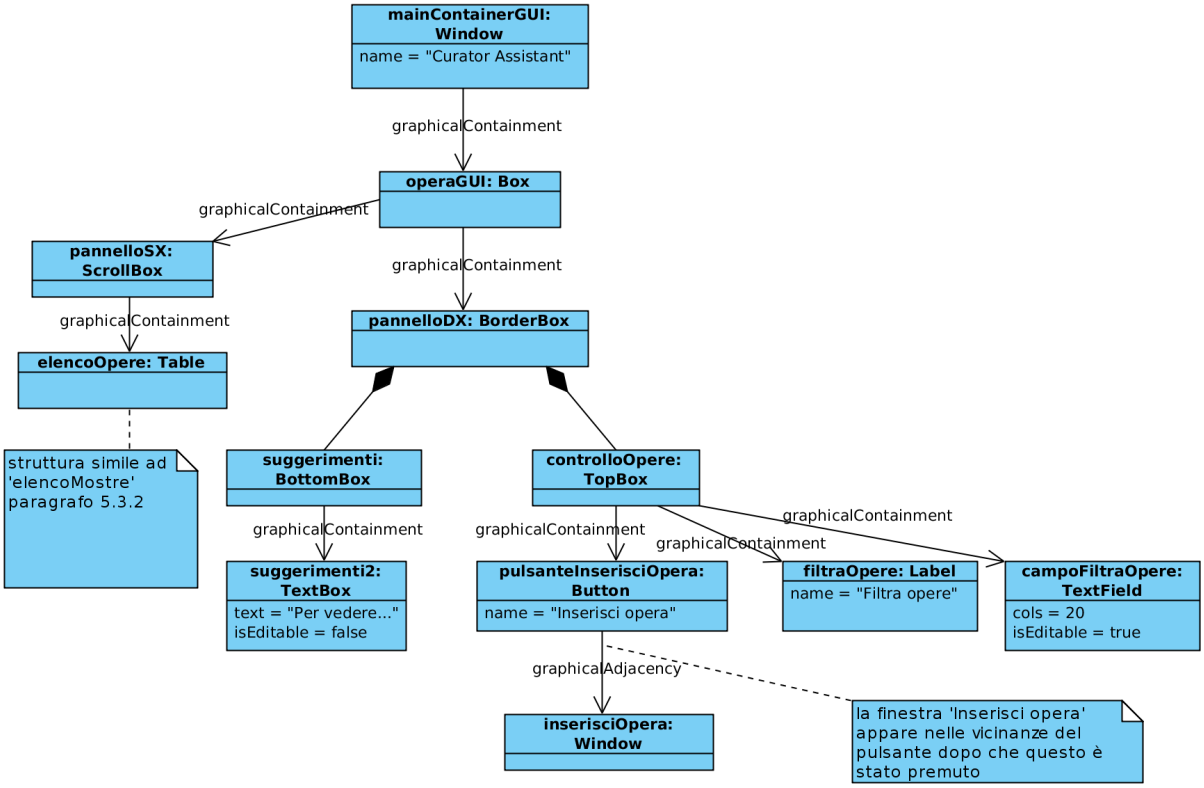
5.4.1 Progettazione astratta "Catalogare nuova opera"

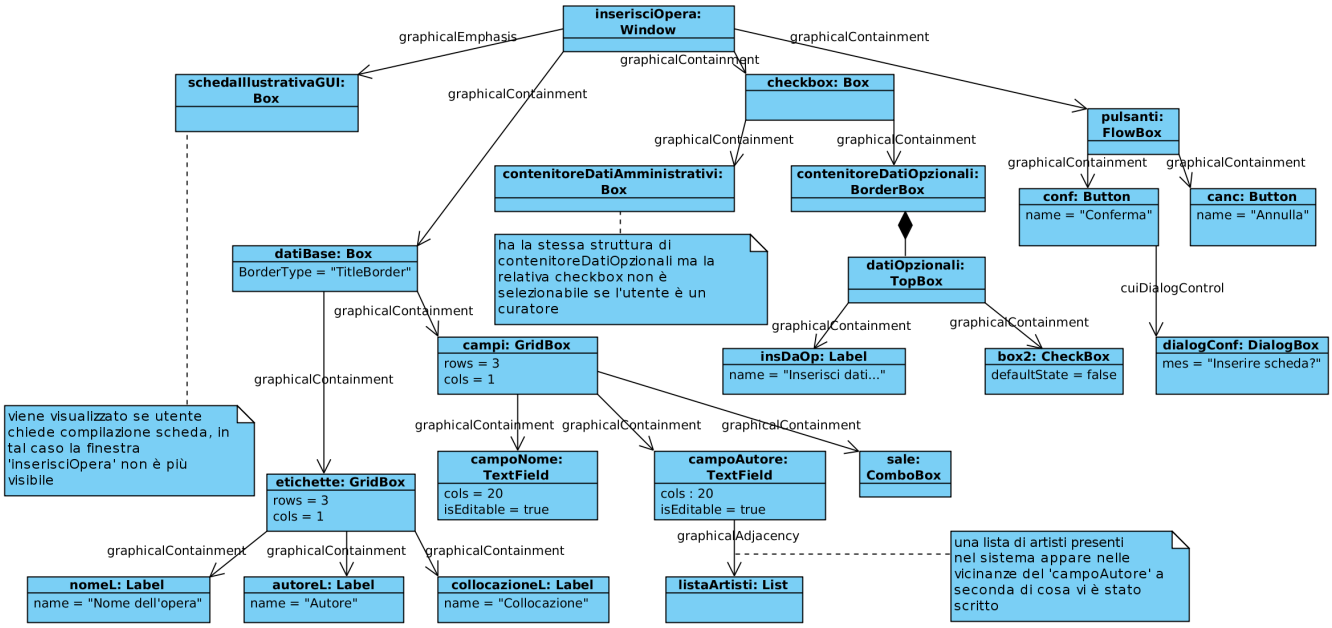



mainContainer: abstractContainer
name = "Curatore Assistant"
absractContainment
contenitoreCentrale: abstractContainer
absractContainment
absractContainment
contenitore1: abstractContainer
contenitore2: abstractContainer
absractContainment
absractContainment
elencoOpere: abstractContainer
inserisciOpera: abastractIndividualComponent
facetInserisciOpera: control
event = "Interagito con 'inserisciOpera' "
action = "Presenta container 'inserisciOpera' "

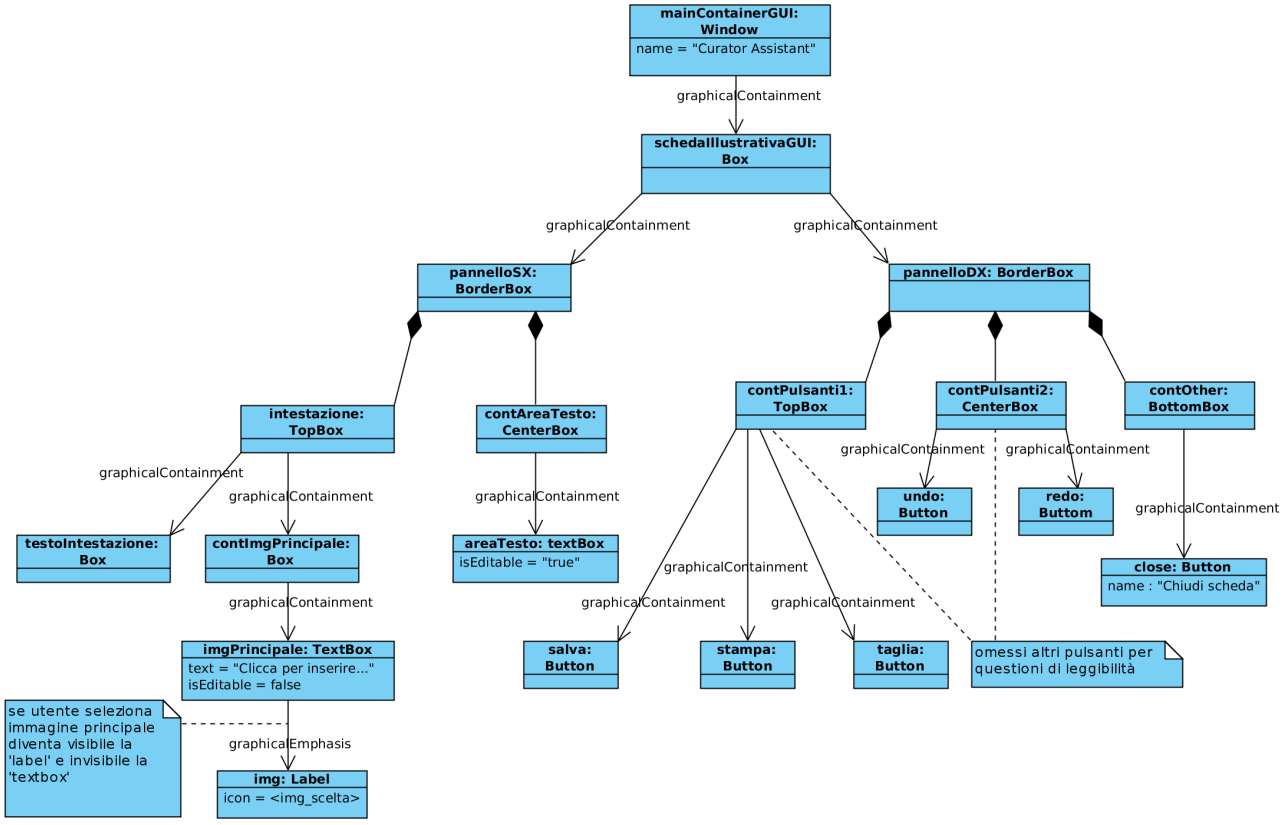

inserisciOpera: abstractContainer
name = "Inserisci opera"
mutualEmphasis
schedaIllustrativa: abstractContainer
absractContainment
absractContainment
absractContainment
datiBase: abstractContainer
name = "Dati base"
contenitoreDatiOpzionali: abstractContainer
name = "Inserisci dati opzionali"
conferma: abastractIndividualComponent
absractContainment
viene presentato se utente chiede compilazione scheda, in tal caso il container 'inserisciOpera' non è più presentato
datiOpzionali: abastractIndividualComponent
facetConferma: control
event = "Interagito con 'conferma' "
action = "Chiedi compilazione scheda illustrativa"
facetDatiOpzionali: control
event = "Interagito con 'datiOpzionali' "
action = "Aggiungi componenti al container"

5.4.2 Progettazione concreta “Catalogare nuova opera”





Se dopo la conferma dell'inserimento di una nuova opera l'utente sceglie di compilare la relativa scheda illustrativa verrà mostrata la pagina di editing per la scheda illustrativa all'interno del "Contenitore principale". Tale pagina avrà dunque la struttura vista in precedenza con 'pannelloSX' e 'pannelloDX'.



5.4.3 Implementazione fisica "Catalogare nuova opera"

Completare il 'Frammento omesso 1' del codice 5.1.3 con il seguente frammento:

```
final SchedaIllustrativaGUI panScheda;
final OperaGUI panOpera;
```

Completare il 'Frammento omesso 2' del codice 5.1.3 con il seguente frammento, oltre a quanto già aggiunto nei paragrafi precedenti:

```
panOpera = new OperaGUI(this);
pulsante1.addActionListener(new AscoltatorePulsanti(this, panOpera,pulsante1));

panScheda = new SchedaIllustrativaGUI(this);
```

Completare il 'Frammento omesso 4' del codice 5.3.3 con il seguente frammento:

```
if (e.getActionCommand().compareTo("Opere")==0) {

                  fin.remove(fin.pannelloSX);
                  fin.remove(fin.pannelloDX);

                  OperaGUI panOpera;
                  panOpera=(OperaGUI)ob;
                  panOpera.ridisegna();

                  fin.add(fin.pannelloSX, BorderLayout.WEST);
                  fin.add(fin.pannelloDX,BorderLayout.EAST);

                  fin.repaint();
                  fin.setVisible(true);

}
```

Aggiungere alla classe 'MainContainerGUI' il seguente metodo:

```
public void visualizza(String s, String s2){

         this.remove(this.pannelloSX);
         this.remove(this.pannelloDX);

         if(s=="Scheda") panScheda.ridisegna(s2);
         if(s=="Opera") panOpera.ridisegna();
         //--------------------------Frammento omesso 5----------------------------------

         //-------------------------------------------------------------------------------
         this.add(this.pannelloSX, BorderLayout.WEST);
         this.add(this.pannelloDX,BorderLayout.EAST);

         this.repaint();
         this.setVisible(true);


}
```

Aggiungere al package la classe 'OperaGUI':

```
package curatorAssistantGUI_betaVs;
import javax.swing.*;
import javax.swing.event.CaretEvent;
import javax.swing.event.CaretListener;
import javax.swing.table.DefaultTableModel;
import javax.swing.text.Caret;
import java.awt.*;
import java.awt.event.*;
import java.io.*;

public class OperaGUI {

         Container panDX;
         Container panSX;
         MainContainerGUI menuPrincipale;

         public OperaGUI(MainContainerGUI fin){


                  panDX = new JPanel();
                  menuPrincipale=fin;

                  DefaultTableModel mod = new DefaultTableModel();
                 mod.addColumn("Identificativo");
                 mod.addColumn("Nome");
                 mod.addColumn("Autore");
                 mod.addColumn("Collocazione");
```

```
        mod.setRowCount(100);

        mod.setValueAt("OP001", 0, 0);
        mod.setValueAt("La persistenza della memoria", 0, 1);
        mod.setValueAt("Salvador Dalì", 0, 2);
        mod.setValueAt("Sala 1", 0, 3);

        JTable tab = new JTable(mod){
        @Override        public boolean isCellEditable(int row, int col) {
                                        return false;
                                }
                };
        //--------------------------Frammento omesso 6----------------------------------

        //------------------------------------------------------------------------------
        panSX = new JScrollPane(tab);
        panSX.setPreferredSize(new Dimension(1000,300));
        JPanel panControlloOpere = new JPanel();
        JPanel contenitorePulsante =new JPanel();
        JPanel contenitoreCampoTesto =new JPanel();
        JPanel suggerimenti =new JPanel();

        panDX.setPreferredSize(new Dimension(300,100));

        panControlloOpere.setLayout(new GridLayout(12,1));

        Image i=java.awt.Toolkit.getDefaultToolkit().getImage(
        "/home/macash/Desktop/CollegamentoIUM/img/plus2.gif");
        i=i.getScaledInstance(15,15,Image.SCALE_DEFAULT);
        ImageIcon ic = new ImageIcon(i);


        JButton pulsInsertOpera = new JButton(" Inserisci nuova opera");
        pulsInsertOpera.setIcon(ic);
        pulsInsertOpera.addActionListener(new MyInsOperaActionListener(menuPrincipale, tab));

        JLabel lab = new JLabel();
        panControlloOpere.add(lab);
        contenitorePulsante.add(pulsInsertOpera);
        panControlloOpere.add(contenitorePulsante);

        panControlloOpere.add(new JLabel());
        panControlloOpere.add(new JLabel("Filtra Opere", JLabel.CENTER));
        JTextField t = new JTextField(20);
        contenitoreCampoTesto.add(t);
        panControlloOpere.add(contenitoreCampoTesto);

        JTextPane tp = new JTextPane();
        tp.setFont(new Font(tp.getFont().getName(), tp.getFont().getStyle(), 11 ));
String s1 = new String("Suggerimenti: \n- Per vedere i dati estesi di un'opera\n fai doppio click ");
String s2 = new String("sulla riga corrispondente.\n Nei dati estesi trovi nuove funzioni come:");
String s3 = new String("\n MODIFICA OPERA \n VEDI SCHEDA ILLUSTRATIVA\n\n -Per filtrare le opere scrivi");
tp.setText(s1+s2+s3+" una parola\n nel campo \"Filtra opere\" e premi INVIO.");
        tp.setEditable(false);
        tp.setBackground(panDX.getBackground());
        suggerimenti.add(tp);
        suggerimenti.setBorder(BorderFactory.createLoweredBevelBorder());

        panDX.add(panControlloOpere, BorderLayout.NORTH);
        panDX.add(suggerimenti, BorderLayout.SOUTH);

        }

        public void ridisegna(){

                menuPrincipale.pannelloSX=panSX;
                menuPrincipale.pannelloDX=panDX;

        }



}
```

Aggiungere in coda alla classe 'OperaGUI' le seguenti classi:

```
class MyInsOperaActionListener implements ActionListener{

        MainContainerGUI menuPrincipale;

        JTable tab;
        JComboBox[] vectorCB = new JComboBox[6];

        JButton conf, canc;
        JFrame inserisciOpera;

        JTextField campoNome = new JTextField(20);
        JTextField campoAutore = new JTextField(20);

        Box gruppo = Box.createHorizontalBox();

        GridLayout gl =new GridLayout(5,1);
        JPanel etichette = new JPanel(gl);
        JPanel etichette2 = new JPanel(gl);
        JLabel nomeL = new JLabel(" Nome dell'opera:"+" "+" "+" "+" "+" "+" ");
        JLabel autoreL = new JLabel(" Autore:");
        JLabel collocazioneL = new JLabel(" Collocazione:");
        JLabel tipologiaL = new JLabel(" Tipologia:");

        JPanel campi = new JPanel(gl);
        JPanel text1 = new JPanel();
        JPanel text2 = new JPanel();

        DefaultComboBoxModel model1 = new DefaultComboBoxModel();
        JComboBox tipo = new JComboBox(model1);
        JPanel tipologia = new JPanel(new FlowLayout(FlowLayout.LEFT));

        DefaultComboBoxModel model1_2 = new DefaultComboBoxModel();
        JComboBox sale = new JComboBox(model1_2);
        JPanel collocazione = new JPanel(new FlowLayout(FlowLayout.LEFT));

        boolean sug=false;
        boolean b=false;

        MyFocusListener mfl = new MyFocusListener(this,true);

        public MyInsOperaActionListener(MainContainerGUI mP , JTable t){tab=t; menuPrincipale=mP;}

        public void actionPerformed(ActionEvent e){

                inserisciOpera = new JFrame("Inserisci opera");

                text1.add(campoNome);
                text2.add(campoAutore);

                model1.addElement("Quadro");
                model1.addElement("Scultura");
                model1.addElement("Altro");
                tipologia.add(tipo);

                model1_2.addElement("Sala1");
                model1_2.addElement("Sala2 ");
                model1_2.addElement("Sala3");
                model1_2.addElement("Sala4");
                model1_2.addElement("Sala5");
                model1_2.addElement("Sala6");

                collocazione.add(sale);

                etichette.add(nomeL);
                etichette.add(autoreL);
                etichette.add(collocazioneL);
                etichette.add(tipologiaL);

                campi.add(text1);
                campi.add(text2);
                campi.add(collocazione);
                campi.add(tipologia);

                gruppo.add(etichette);
                gruppo.add(campi);
                gruppo.setBorder(BorderFactory.createTitledBorder("DATI BASE"));
```

```
            JPanel datiBase = new JPanel(new FlowLayout(FlowLayout.LEFT));
            datiBase.add(gruppo);

            JPanel contenitoreDatiAmministrativi = new JPanel(new BorderLayout());
            JPanel DatiAmministrativi = new JPanel(new FlowLayout(FlowLayout.LEFT));
            JLabel insOp = new JLabel("Inserisci dati amministrativi: "+" ");
            insOp.setForeground(new Color(150,150,150));
            JCheckBox box1 = new JCheckBox();
            box1.setEnabled(false);

            DatiAmministrativi.add(insOp);
            DatiAmministrativi.add(box1);
            contenitoreDatiAmministrativi.add(DatiAmministrativi, BorderLayout.NORTH);

            JPanel contenitoreDatiOpzionali = new JPanel(new BorderLayout());
            JPanel datiOpzionali = new JPanel(new FlowLayout(FlowLayout.LEFT));
            JLabel datiOpt = new JLabel("Inserisci dati opzionali: "+" ");
            JCheckBox box2 = new JCheckBox();
            datiOpzionali.add(datiOpt);
            datiOpzionali.add(box2);
            contenitoreDatiOpzionali.add(datiOpzionali, BorderLayout.NORTH);

            JPanel checkbox = new JPanel(new FlowLayout(FlowLayout.LEFT));
            checkbox.add(contenitoreDatiAmministrativi, BorderLayout.NORTH);
            checkbox.add(contenitoreDatiOpzionali, BorderLayout.SOUTH);

            JPanel pulsanti = new JPanel(new FlowLayout());
            canc = new JButton("Annulla");
            conf = new JButton("Conferma");
            conf.addActionListener(new MyConfActionListener2(this, menuPrincipale));

            pulsanti.add(canc);
            pulsanti.add(conf);

            conf.setEnabled(false);
            campoNome.addKeyListener(new MyKeyListener2(this, campoNome,conf));
            campoAutore.addKeyListener(new MyKeyListener1(this,campoAutore,conf));

            sale.addFocusListener(mfl);

            inserisciOpera.add(datiBase,BorderLayout.NORTH);
            inserisciOpera.add(checkbox,BorderLayout.CENTER);
            inserisciOpera.add(pulsanti,BorderLayout.SOUTH);

            inserisciOpera.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
            inserisciOpera.pack();
            inserisciOpera.setSize(420,400);
            inserisciOpera.setVisible(true);

        }

        public void riDisegnaDatiBase( boolean flag, boolean focus  ){

            if (flag==true) {

                sug=true;
                gl.setRows(7);

                gruppo.remove(etichette);
                gruppo.remove(campi);

                etichette.removeAll();

                etichette.add(nomeL);
                etichette.add(autoreL);
                etichette.add(new JLabel(""));
                etichette.add(new JLabel(""));
                etichette.add(collocazioneL);
                etichette.add(tipologiaL);

                campi.removeAll();

                DefaultListModel listmod1 = new DefaultListModel();
                listmod1.addElement("Dalì Salvador");
                JList list= new JList(listmod1);

                campi.add(text1);
                campi.add(text2);
                campi.add(list);
```

```
                        campi.add(new JLabel(""));
                        campi.add(collocazione);
                        campi.add(tipologia);

                        gruppo.add(etichette);
                        gruppo.add(campi);

                        text2.setBackground(new Color(255,255,255));

                        list.addMouseListener(new MouseAdapter(){

                                public void mousePressed(MouseEvent e){

                                Object elemSel =selezionaElemento((JList)e.getSource(),e.getPoint());

                                campoAutore.setText(elemSel.toString());

                                if(campoAutore.getText().compareTo("Dalì Salvador")==0 && !
campoNome.getText().isEmpty()){conf.setEnabled(true);}

                                }

                        });

                        inserisciOpera.setSize(420,500);
                        inserisciOpera.repaint();
                        inserisciOpera.setVisible(true);

                        if (focus==true) {campoAutore.requestFocus();}


                }
                else {

                        if (sug==true){

                                sug=false;
                                gl.setRows(5);

                                gruppo.remove(etichette);
                                gruppo.remove(campi);

                                etichette.removeAll();
                                etichette.add(nomeL);
                                etichette.add(autoreL);
                                etichette.add(collocazioneL);
                                etichette.add(tipologiaL);

                                campi.removeAll();
                                campi.add(text1);
                                campi.add(text2);
                                campi.add(collocazione);
                                campi.add(tipologia);

                                gruppo.add(etichette);
                                gruppo.add(campi);

                                text2.setBackground(text1.getBackground());

                                if (focus==true) {
                                        campoAutore.requestFocus();
                                        inserisciOpera.setSize(420,500);
                                }
                                else {

                                        if(this.segnalaUtente()==1){ inserisciOpera.setSize(655,500);}

                                }

                                inserisciOpera.repaint();
                                inserisciOpera.setVisible(true);

                        }
                        else {
                                if(sug==false && focus==false){
                                        this.segnalaUtente();
                                        if(this.segnalaUtente()==1){ inserisciOpera.setSize(655,500);}
                                }
```

```
                        }

                }

        }//end ridisegna

        public static Object selezionaElemento(JList l, Point clickPoint){

                int index= l.locationToIndex(clickPoint);
                if (index >=0) {
                        l.setSelectedIndex(index);
                        Object selezionato= l.getModel().getElementAt(index);
                        return selezionato;
                }
                else return null;

        }

        private int segnalaUtente(){

                if(!campoAutore.getText().isEmpty() && campoAutore.getText().compareTo("dalì")!=0 &&
campoAutore.getText().compareTo("Dalì Salvador")!=0) {

                                if(b==false){
                                        Label l=new Label("L'artista non è presente nel sistema!");
                                        l.setForeground(new Color(231,25,6));

                                        etichette2.add(new Label(""));
                                        etichette2.add(l);
                                        etichette2.add(new Label(""));
                                        etichette2.add(new Label(""));
                                        gruppo.add(etichette2);
                                        b=true;
                                }
                                else {gruppo.add(etichette2);}
            return 1;
                }

                campoAutore.setText("Dalì Salvador");
                return 0;

        }

}
```

```
class MyKeyListener1 implements KeyListener {

        JTextField tx;
        MyInsOperaActionListener insOp;

        MyKeyListener1(MyInsOperaActionListener iO, JTextField t, JButton b){ tx=t; insOp=iO;  }


                        public void keyPressed(KeyEvent e){}
                        public void keyTyped(KeyEvent e){}
                        public void keyReleased(KeyEvent e){

                        if(tx.getText().compareTo("Dalì Salvador")==0 && !insOp.campoNome.getText().isEmpty()){
                       insOp.conf.setEnabled(true);}
                                else insOp.conf.setEnabled(false);

                                insOp.gruppo.remove(insOp.etichette2);
                                insOp.inserisciOpera.setSize(420,500);


                                if(tx.getText().compareTo("d")==0 ||tx.getText().compareTo("da")==0 ||
tx.getText().compareTo("dal")==0 ||tx.getText().compareTo("dalì")==0 ) {


                                insOp.riDisegnaDatiBase(true,true);

                                }
                                else insOp.riDisegnaDatiBase(false,true);

                        }
}
```

```
class MyFocusListener implements FocusListener{

        MyInsOperaActionListener insOp;
        boolean focus,flag;

        MyFocusListener(MyInsOperaActionListener iO, boolean f){ insOp=iO; flag=f;}

        public void focusGained(FocusEvent arg0) {

                if(insOp.campoAutore.getText().compareTo("Dalì Salvador")==0 && !
insOp.campoNome.getText().isEmpty()){insOp.conf.setEnabled(true);}
                insOp.riDisegnaDatiBase(false,false);
                focus=true;

        }

        public void focusLost(FocusEvent arg0) {

                if(insOp.campoAutore.getText().compareTo("Dalì Salvador")==0 && !
insOp.campoNome.getText().isEmpty()){insOp.conf.setEnabled(true);}
                focus=false;

        }

        public boolean getFocus(){ return focus; }

}
```

```
class MyKeyListener2 implements KeyListener {

        JTextField tx;
        MyInsOperaActionListener insOp;

        MyKeyListener2(MyInsOperaActionListener iO, JTextField t, JButton b){ tx=t;insOp=iO;}

                        public void keyPressed(KeyEvent e){}
                        public void keyTyped(KeyEvent e){}
                        public void keyReleased(KeyEvent e){

                                if(insOp.campoAutore.getText().compareTo("Dalì Salvador")==0 && !
tx.getText().isEmpty()){insOp.conf.setEnabled(true);}
                                else insOp.conf.setEnabled(false);

                        }
}
```

```
class MyConfActionListener2 implements ActionListener {

        MainContainerGUI menuPrincipale;
        MyInsOperaActionListener insOp;

        public MyConfActionListener2(MyInsOperaActionListener iO, MainContainerGUI mP){insOp=iO;
menuPrincipale=mP;}

        public void actionPerformed(ActionEvent e){

                insOp.tab.getModel().setValueAt("OP002", 1, 0);
                insOp.tab.getModel().setValueAt(insOp.campoNome.getText(), 1, 1);
                insOp.tab.getModel().setValueAt(insOp.campoAutore.getText(), 1, 2);
                int index=insOp.sale.getSelectedIndex();
                String s = insOp.sale.getModel().getElementAt(index).toString();
                insOp.tab.getModel().setValueAt(s, 1, 3);
                insOp.inserisciOpera.dispose();
                int risp = JOptionPane.showConfirmDialog(insOp.inserisciOpera,
               "Inserita una nuova opera \n\nVuoi inserire la scheda illustrativa dell'opera?",
               "Messaggio",JOptionPane.YES_NO_OPTION);
                if (risp==JOptionPane.YES_OPTION) {

                        menuPrincipale.visualizza("Scheda", insOp.campoNome.getText());

                }

        }
}
```

Aggiungere al package la classe 'SchedaIllustrativaGUI':

```
package curatorAssistantGUI_betaVs;
import javax.swing.*;
import java.awt.*;
import javax.swing.text.*;
import javax.swing.tree.*;
import java.awt.Toolkit;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.io.File;

public class SchedaIllustrativaGUI {

        JPanel panSX, panDX;
        MainContainerGUI menuPrincipale;
        MyMouseListener mml = new MyMouseListener(this);
        JPanel contImgPrincipale;
        JButton salva,undo;
        JTextField tf1;
        final JTextPane areaTesto;

        public SchedaIllustrativaGUI(MainContainerGUI fin){

                panSX = new JPanel(new BorderLayout());
                panDX = new JPanel(new BorderLayout());
                menuPrincipale=fin;
                JPanel contAreaTesto = new JPanel();
                areaTesto = new JTextPane();
                areaTesto.setPreferredSize(new Dimension(1000,1000));
                areaTesto.setBorder(BorderFactory.createRaisedBevelBorder());
                contAreaTesto.setBorder(BorderFactory.createTitledBorder("Descrizione della scheda:"));
                contAreaTesto.add(areaTesto);
                areaTesto.addKeyListener(new KeyListener() {

                        public void keyPressed(KeyEvent arg0) {}

                        public void keyReleased(KeyEvent arg0) {

                        if(!areaTesto.getText().isEmpty()){salva.setEnabled(true); undo.setEnabled(true);}
                        else {salva.setEnabled(false); undo.setEnabled(false);}

                        }

                        public void keyTyped(KeyEvent arg0) {

                        }});

                contImgPrincipale = new JPanel();
                JPanel intestazione = new JPanel(new BorderLayout());
                JPanel testoIntestazione = new JPanel(new FlowLayout(FlowLayout.LEFT));
                JTextPane imgPrincipale = new JTextPane();
                imgPrincipale.setPreferredSize(new Dimension(200,200));
                imgPrincipale.setBorder(BorderFactory.createLineBorder(new Color(0,0,0)));
                imgPrincipale.setText("\n\n\n\n\n<clicca per inserire immagine>");
                imgPrincipale.setEditable(false);
                imgPrincipale.setName("Inserisci immagine principale");
                imgPrincipale.addMouseListener(mml);

                JPanel labels1 = new JPanel (new GridLayout(13,1));
                labels1.add(new JLabel(""));
                labels1.add(new JLabel("Titolo dell'opera:"));
                labels1.add(new JLabel(""));
                labels1.add(new JLabel("Autore dell'opera:"));
                labels1.add(new JLabel(""));

                JPanel labels2 = new JPanel (new GridLayout(13,1));
                labels2.add(new JLabel(""));
                tf1=new JTextField();
                tf1.setEditable(false);
                tf1.setBorder(BorderFactory.createEmptyBorder());
                labels2.add(tf1);
                labels2.add(new JLabel(""));
                JTextField tf2=new JTextField(" Dalì Salvador");
```

```
tf2.setEditable(false);
tf2.setBorder(BorderFactory.createEmptyBorder());
labels2.add(tf2);
labels2.add(new JLabel(""));

Box group = Box.createHorizontalBox();
group.add(labels1);
group.add(labels2);

testoIntestazione.add(group);
contImgPrincipale.add(imgPrincipale);

intestazione.add(testoIntestazione, BorderLayout.CENTER);
intestazione.add(contImgPrincipale, BorderLayout.WEST);

panSX.add(intestazione, BorderLayout.NORTH);
panSX.add(contAreaTesto, BorderLayout.CENTER);

panSX.setBorder(BorderFactory.createEtchedBorder());

Image i = Toolkit.getDefaultToolkit().createImage(
"/home/macash/Desktop/CollegamentoIUM/img/editor/New.png");

i=i.getScaledInstance(30,30,Image.SCALE_DEFAULT);
ImageIcon ic = new ImageIcon(i);

JButton nuovo = new JButton();
nuovo.setIcon(ic);
nuovo.setPreferredSize(new Dimension(30,30));
nuovo.setBackground(new Color(238,238,238));

Image i2 = Toolkit.getDefaultToolkit().createImage(
"/home/macash/Desktop/CollegamentoIUM/img/editor/Open3.png");
i2=i2.getScaledInstance(30,30,Image.SCALE_DEFAULT);
ImageIcon ic2 = new ImageIcon(i2);

JButton apri = new JButton();
apri.setIcon(ic2);
apri.setPreferredSize(new Dimension(30,30));
apri.setBackground(new Color(238,238,238));

Image i3 = Toolkit.getDefaultToolkit().createImage(
"/home/macash/Desktop/CollegamentoIUM/img/editor/Save.png");
i3=i3.getScaledInstance(30,30,Image.SCALE_DEFAULT);
ImageIcon ic3 = new ImageIcon(i3);

salva = new JButton();
salva.setIcon(ic3);
salva.setPreferredSize(new Dimension(30,30));
salva.setBackground(new Color(238,238,238));

Image i4 = Toolkit.getDefaultToolkit().createImage(
"/home/macash/Desktop/CollegamentoIUM/img/editor/Print2.png");
i4=i4.getScaledInstance(30,30,Image.SCALE_DEFAULT);
ImageIcon ic4 = new ImageIcon(i4);

JButton stampa = new JButton();
stampa.setIcon(ic4);
stampa.setPreferredSize(new Dimension(30,30));
stampa.setBackground(new Color(238,238,238));

Image i5 = Toolkit.getDefaultToolkit().createImage(
"/home/macash/Desktop/CollegamentoIUM/img/editor/cut.png");
i5=i5.getScaledInstance(30,30,Image.SCALE_DEFAULT);
ImageIcon ic5 = new ImageIcon(i5);

JButton taglia = new JButton();
taglia.setIcon(ic5);
taglia.setPreferredSize(new Dimension(30,30));
taglia.setBackground(new Color(238,238,238));

Image i6 = Toolkit.getDefaultToolkit().createImage(
"/home/macash/Desktop/CollegamentoIUM/img/editor/copy2.png");
i6=i6.getScaledInstance(30,30,Image.SCALE_DEFAULT);
ImageIcon ic6 = new ImageIcon(i6);

JButton copia = new JButton();
copia.setIcon(ic6);
copia.setPreferredSize(new Dimension(30,30));
copia.setBackground(new Color(238,238,238));
```

```
Image i7 = Toolkit.getDefaultToolkit().createImage(
"/home/macash/Desktop/CollegamentoIUM/img/editor/Paste.png");
i7=i7.getScaledInstance(30,30,Image.SCALE_DEFAULT);
ImageIcon ic7 = new ImageIcon(i7);

JButton incolla = new JButton();
incolla.setIcon(ic7);
incolla.setPreferredSize(new Dimension(30,30));
incolla.setBackground(new Color(238,238,238));

Image i8 = Toolkit.getDefaultToolkit().createImage(
"/home/macash/Desktop/CollegamentoIUM/img/editor/Undo.png");
i8=i8.getScaledInstance(30,30,Image.SCALE_DEFAULT);
ImageIcon ic8 = new ImageIcon(i8);

undo = new JButton();
undo.setIcon(ic8);
undo.setPreferredSize(new Dimension(30,30));
undo.setBackground(new Color(238,238,238));
undo.addActionListener(new ActionListener(){ public void actionPerformed(ActionEvent e){
areaTesto.setText(""); undo.setEnabled(false);} });

Image i9 = Toolkit.getDefaultToolkit().createImage(
"/home/macash/Desktop/CollegamentoIUM/img/editor/Redo.png");
i9=i9.getScaledInstance(30,30,Image.SCALE_DEFAULT);
ImageIcon ic9 = new ImageIcon(i9);

JButton redo = new JButton();
redo.setIcon(ic9);
redo.setPreferredSize(new Dimension(30,30));
redo.setBackground(new Color(238,238,238));

Image i10 = Toolkit.getDefaultToolkit().createImage(
"/home/macash/Desktop/CollegamentoIUM/img/editor/bold.png");
i10=i10.getScaledInstance(30,30,Image.SCALE_DEFAULT);
ImageIcon ic10 = new ImageIcon(i10);

JButton bold = new JButton();
bold.setIcon(ic10);
bold.setPreferredSize(new Dimension(30,30));
bold.setBackground(new Color(238,238,238));

Image i11 = Toolkit.getDefaultToolkit().createImage(
"/home/macash/Desktop/CollegamentoIUM/img/editor/italic.png");
i11=i11.getScaledInstance(30,30,Image.SCALE_DEFAULT);
ImageIcon ic11 = new ImageIcon(i11);

JButton italic = new JButton();
italic.setIcon(ic11);
italic.setPreferredSize(new Dimension(30,30));
italic.setBackground(new Color(238,238,238));

Image i12 = Toolkit.getDefaultToolkit().createImage(
"/home/macash/Desktop/CollegamentoIUM/img/editor/alignLeft.png");
i12=i12.getScaledInstance(30,30,Image.SCALE_DEFAULT);
ImageIcon ic12 = new ImageIcon(i12);

JButton alignLeft = new JButton();
alignLeft.setIcon(ic12);
alignLeft.setPreferredSize(new Dimension(30,30));
alignLeft.setBackground(new Color(238,238,238));

Image i13 = Toolkit.getDefaultToolkit().createImage(
"/home/macash/Desktop/CollegamentoIUM/img/editor/alignRight.png");
i13=i13.getScaledInstance(30,30,Image.SCALE_DEFAULT);
ImageIcon ic13 = new ImageIcon(i13);

JButton alignRight = new JButton();
alignRight.setIcon(ic13);
alignRight.setPreferredSize(new Dimension(30,30));
alignRight.setBackground(new Color(238,238,238));

Image i14 = Toolkit.getDefaultToolkit().createImage(
"/home/macash/Desktop/CollegamentoIUM/img/editor/justify.png");
i14=i14.getScaledInstance(30,30,Image.SCALE_DEFAULT);
ImageIcon ic14 = new ImageIcon(i14);

JButton justified = new JButton();
justified.setIcon(ic14);
justified.setPreferredSize(new Dimension(30,30));
```

```
                justified.setBackground(new Color(238,238,238));

                JPanel contPulsanti1 = new JPanel();
                JPanel contPulsanti2 = new JPanel(new FlowLayout(FlowLayout.LEFT));

                salva.setEnabled(false);
                taglia.setEnabled(false);
                copia.setEnabled(false);
                contPulsanti1.add(nuovo);
                contPulsanti1.add(apri);
                contPulsanti1.add(salva);
                contPulsanti1.add(stampa);
                contPulsanti1.add(taglia);
                contPulsanti1.add(copia);
                contPulsanti1.add(incolla);

                undo.setEnabled(false);
                redo.setEnabled(false);
                contPulsanti2.add(undo);
                contPulsanti2.add(redo);
                contPulsanti2.add(bold);
                contPulsanti2.add(italic);
                contPulsanti2.add(alignLeft);
                contPulsanti2.add(alignRight);
                contPulsanti2.add(justified);

                salva.setName("Salva");
                salva.addMouseListener(mml);

                DefaultComboBoxModel model1 = new DefaultComboBoxModel();
                model1.addElement("08");
                model1.addElement("09");
                model1.addElement("10");
                model1.addElement("11");
                model1.addElement("12");
                model1.addElement("14");
                JComboBox size = new JComboBox(model1);

                DefaultComboBoxModel model2 = new DefaultComboBoxModel();
                model2.addElement("Times New Roman");
                model2.addElement("Georgia");
                model2.addElement("Arial");
                model2.addElement("Verdana");
                JComboBox style = new JComboBox(model2);

                JPanel contOther = new JPanel(new BorderLayout());
                JPanel font = new JPanel(new FlowLayout(FlowLayout.LEFT));
                contOther.setPreferredSize(new Dimension(200,540));

                JPanel contClose = new JPanel();
                JButton close = new JButton("Chiudi scheda illustrativa");
                contClose.add(close);
                contClose.setPreferredSize(new Dimension(200,405));

                Image i0=java.awt.Toolkit.getDefaultToolkit().getImage(
               "/home/macash/Desktop/CollegamentoIUM/img/close.gif");
                i0=i0.getScaledInstance(15,15,Image.SCALE_DEFAULT);
                ImageIcon ic0 = new ImageIcon(i0);

                close.setIcon(ic0);
                close.addActionListener(new ActionListener(){

                        public void actionPerformed(ActionEvent e) {
                                if(salva.isEnabled()){

                                        int risp =JOptionPane.showConfirmDialog(menuPrincipale,
"Vuoi salvare la scheda illustrativa?", "Chiudi scheda", JOptionPane.YES_NO_CANCEL_OPTION);
                                        if (risp==JOptionPane.YES_OPTION || risp==JOptionPane.NO_OPTION){
menuPrincipale.visualizza("Opera",null);}

                                }
                                else {menuPrincipale.visualizza("Opera",null);}

                        }

                });

                font.add(size);
                font.add(style);
                contOther.add(font, BorderLayout.NORTH);
```

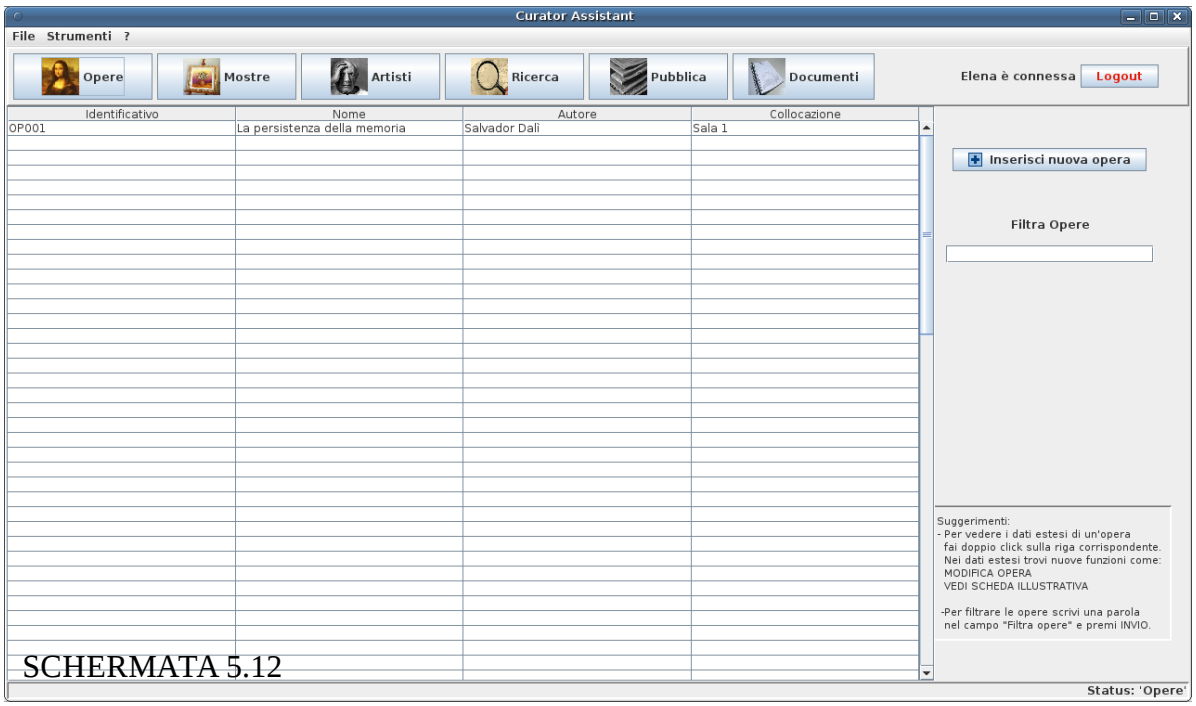
```
                contOther.add(contClose, BorderLayout.SOUTH);

                panDX.add(contPulsanti1, BorderLayout.NORTH);
                panDX.add(contPulsanti2, BorderLayout.CENTER);
                panDX.add(contOther, BorderLayout.SOUTH);

        }

        public void ridisegna(String s){
                tf1.setText(s);
                menuPrincipale.pannelloSX=panSX;
                menuPrincipale.pannelloDX=panDX;

        };


}
```

Aggiungere in coda alla classe 'SchedaIllustrativaGUI' la seguente classe:

```
class MyMouseListener implements MouseListener {

        JPopupMenu popup;
        SchedaIllustrativaGUI siGUI;

        MyMouseListener(SchedaIllustrativaGUI sI){ siGUI=sI;  }

        public void mouseEntered(MouseEvent arg0) {

                JComponent c = (JComponent)arg0.getSource();
                if(c.getName()=="Salva"){
                        popup = new JPopupMenu();
                        popup.add(c.getName());
                        popup.show(c, c.getX()-60, c.getY()+28);

                }
        }

        public void mouseExited(MouseEvent arg0) {

                if (popup!=null) popup.setVisible(false);

        }

        public void mouseClicked(MouseEvent arg0) {

                JComponent c = (JComponent)arg0.getSource();
                if(c.getName()=="Inserisci immagine principale"){

                        JFileChooser fc = new JFileChooser();
                        fc.setDialogTitle("Inserisci immagine");
                        int sel=fc.showOpenDialog(c);
                        if(sel==JFileChooser.APPROVE_OPTION) {

                                File f = fc.getSelectedFile() ;
                                Image i = Toolkit.getDefaultToolkit().createImage(f.getAbsolutePath());
                                i=i.getScaledInstance(200,200,Image.SCALE_DEFAULT);
                                ImageIcon ic = new ImageIcon(i);
                                JLabel img = new JLabel();
                                img.setIcon(ic);
                                siGUI.contImgPrincipale.removeAll();
                                siGUI.contImgPrincipale.add(img);
                                siGUI.salva.setEnabled(true);
                                siGUI.menuPrincipale.setVisible(true);

                        }


                }

        }
        public void mousePressed(MouseEvent arg0) {}
        public void mouseReleased(MouseEvent arg0) {}

}
```

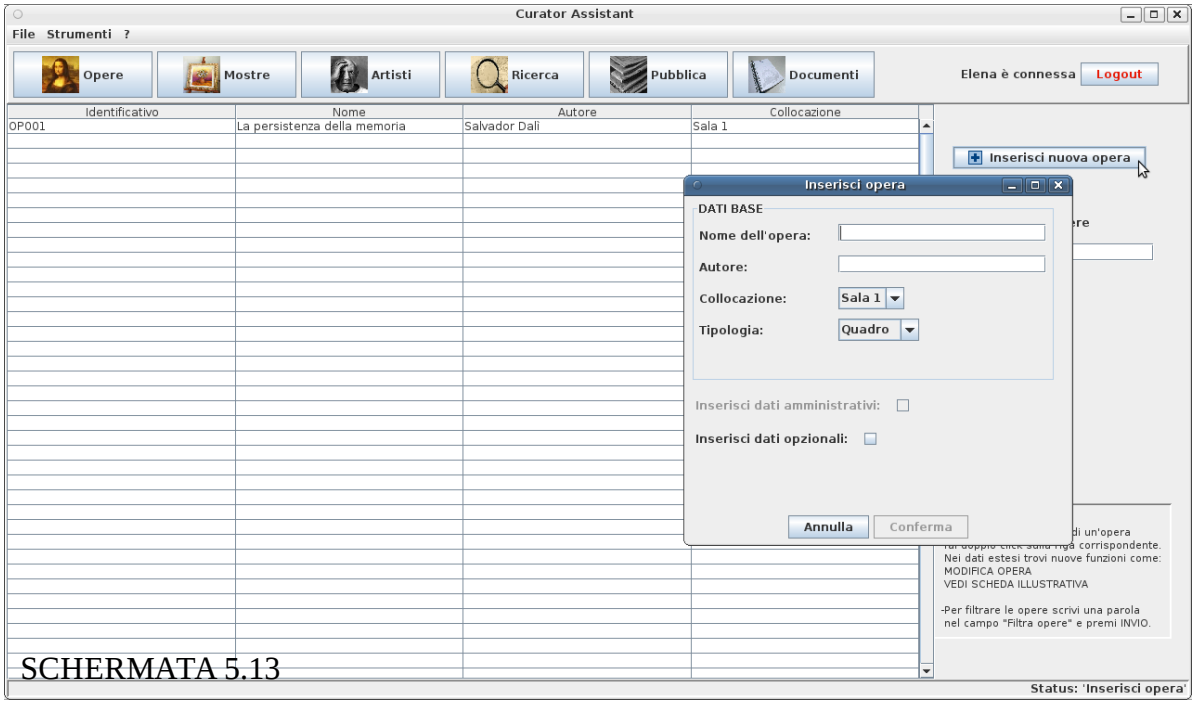
5.4.4 GUI “Catalogare nuova opera”

Elena clicca su 'Opere' dalla 'Pagina iniziale' e visualizza la schermata seguente:



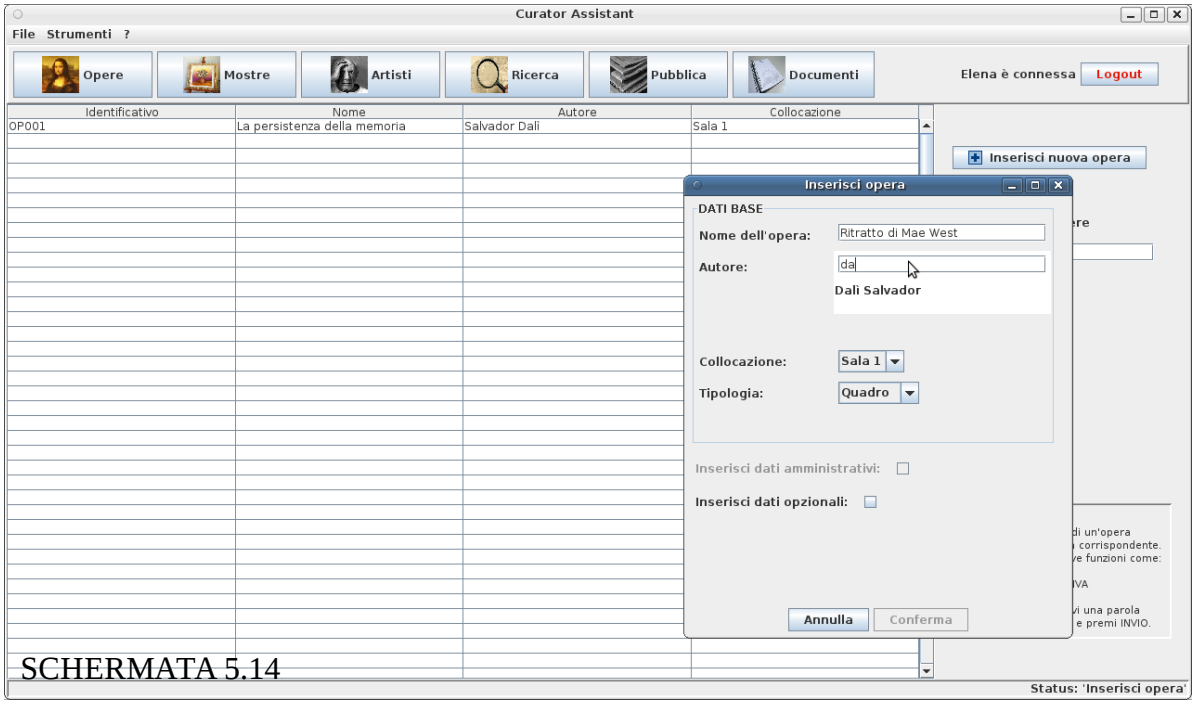
SCHERMATA 5.12

Elena clicca su 'Inserisci nuova opera' :



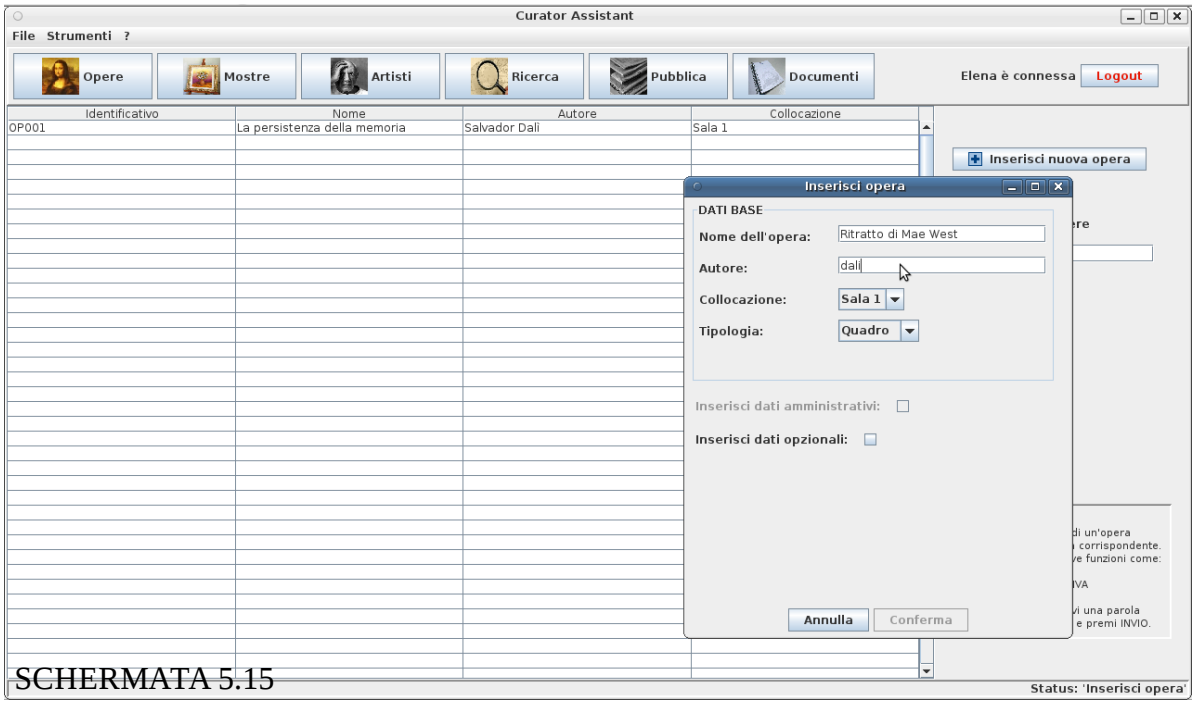
SCHERMATA 5.13

Elena compila i 'Dati base'; mentre scrive il nome dell'autore il sistema visualizza una lista di artisti presenti nel sistema che corrispondono, in tutto o in parte, a quanto scritto:



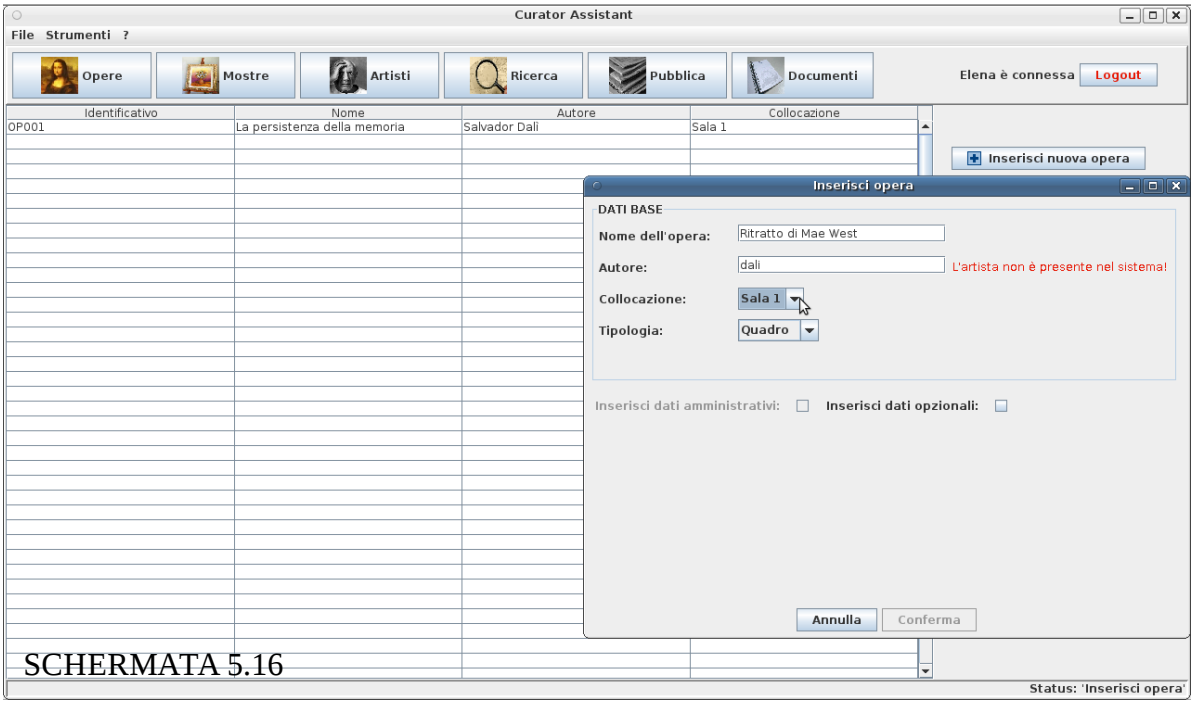
SCHERMATA 5.14

Elena non seleziona l'artista suggerito dalla lista anche se è proprio quello che vuole inserire e continua a digitare il suo nome. Elena scrive ' dali ' anziché ' dalì ' per sbaglio, la stringa non fa più *match* con alcun artista quindi la lista di suggerimenti scompare.



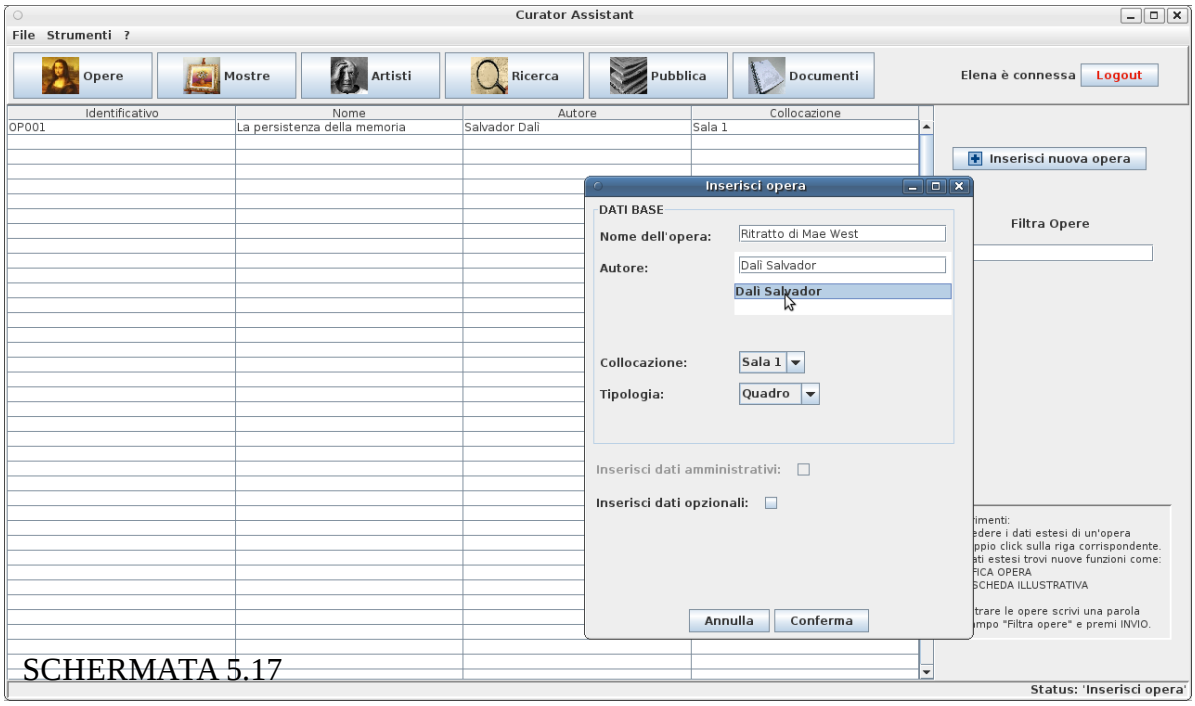
SCHERMATA 5.15

Elena continua la compilazione dei dati ma poiché l'autore digitato non corrisponde a nessun artista presente nel sistema ciò gli viene segnalato tramite un messaggio. Elena non può confermare i dati finché l'opera non è associata ad un artista valido[9].



SCHERMATA 5.16

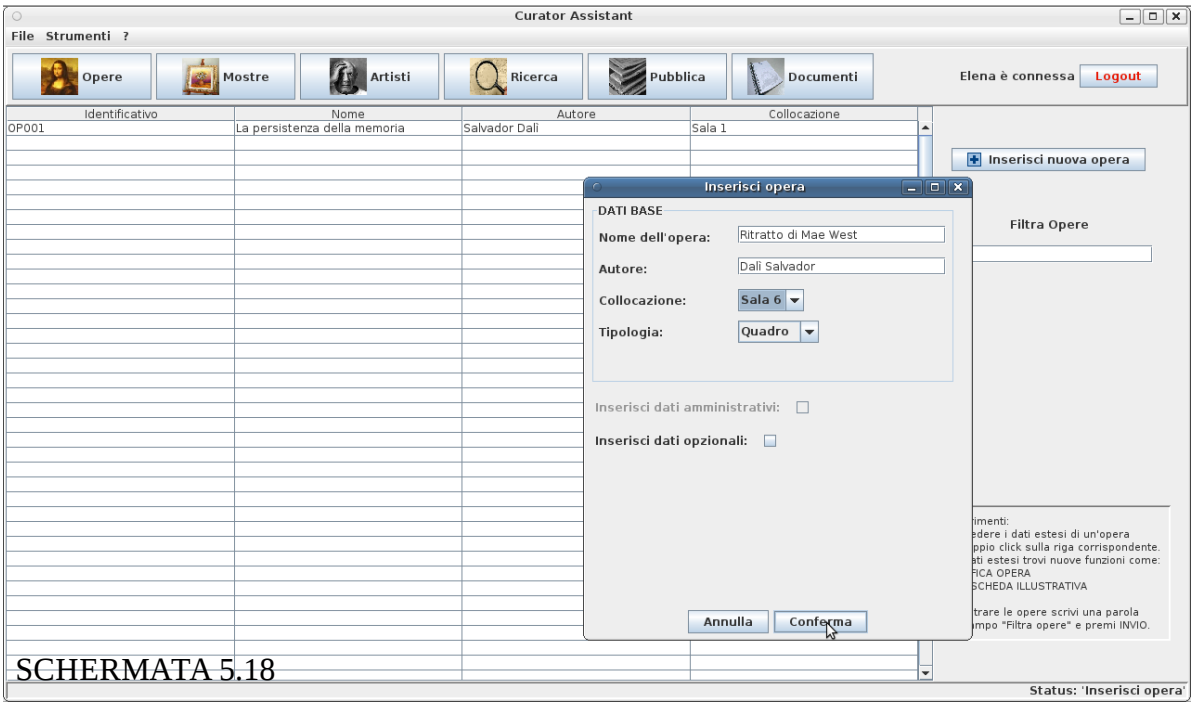
Questa volta Elena ricompila il campo autore selezionando il suggerimento:



SCHERMATA 5.17

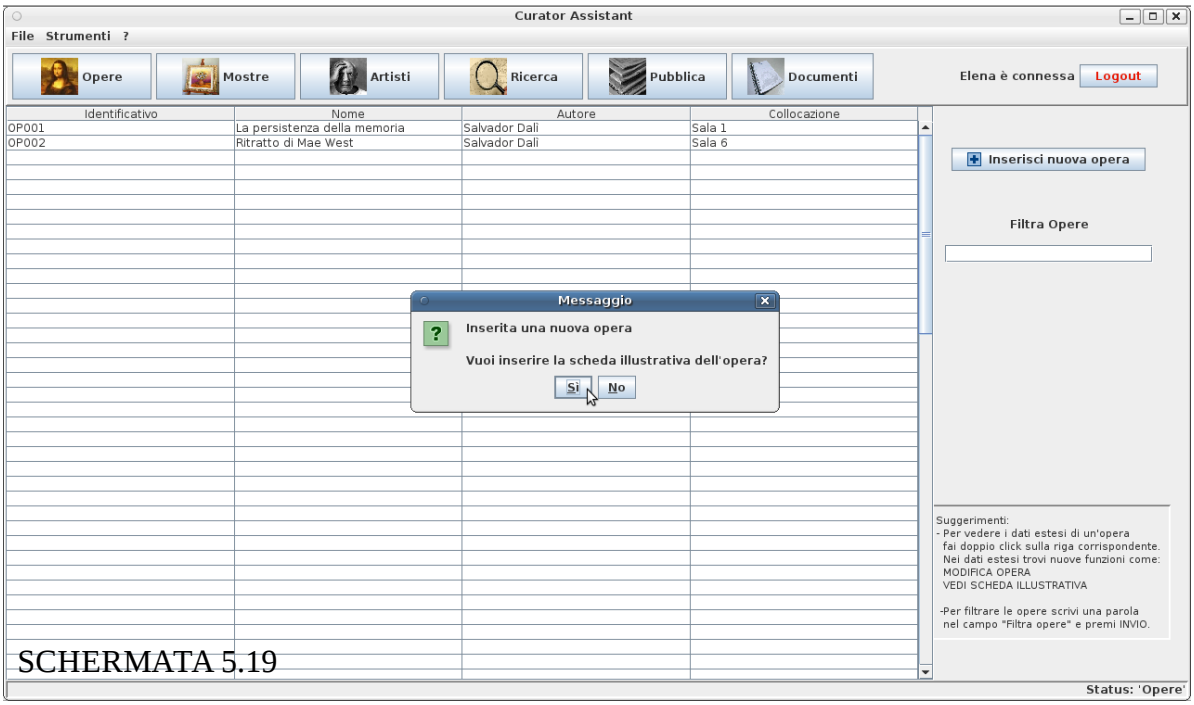
9 Come da specifica **UC_04_01_01** documento **4-CA_UCM_E1_R3**

Ora, compilando anche il resto dei dati, il pulsante 'Conferma' è abilitato:



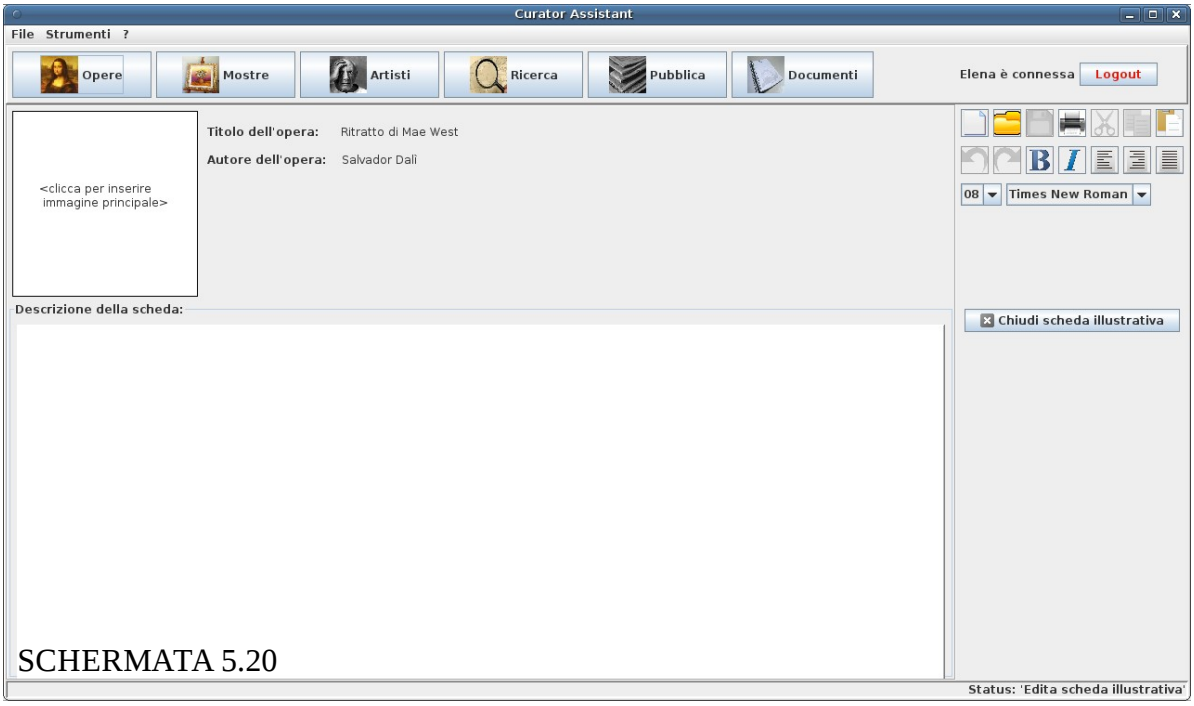
SCHERMATA 5.18

Cliccando su 'Conferma' è presentata una *dialog* che oltre a confermare l'avvenuto successo dell'inserimento, chiede ad Elena se vuole inserire la scheda illustrativa associata:



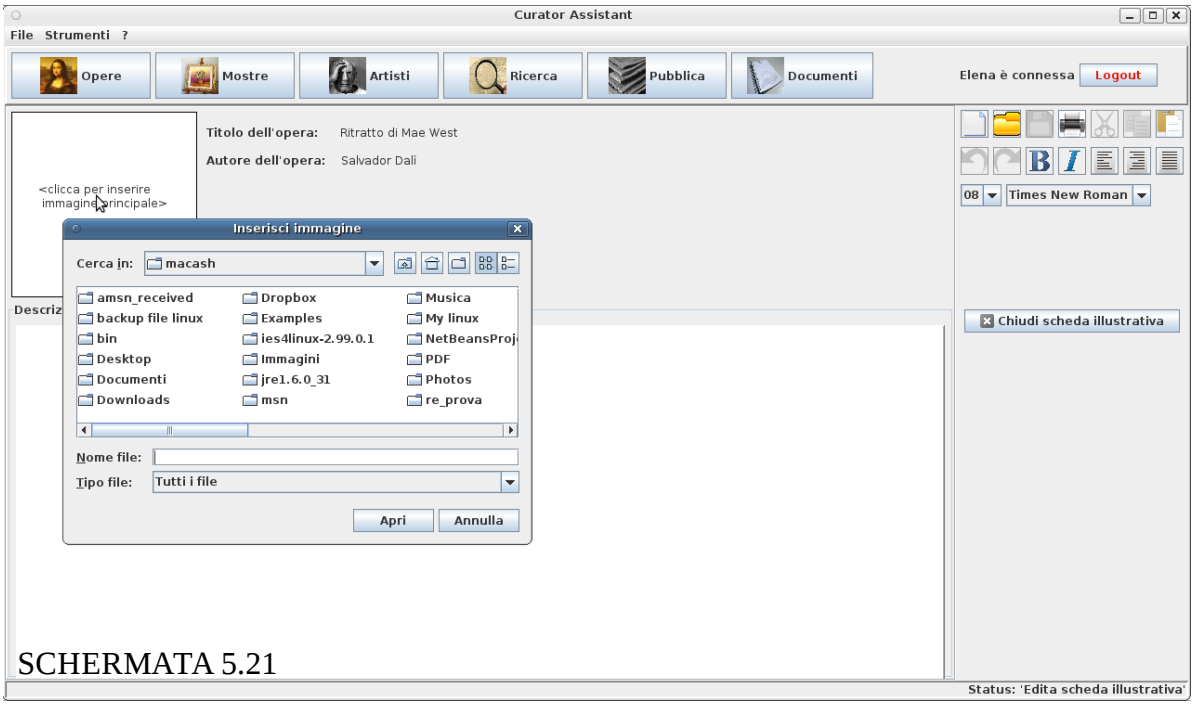
SCHERMATA 5.19

Viene presentato l'*editor* di scheda illustrativa con i dati relativi all'opera appena inserita:



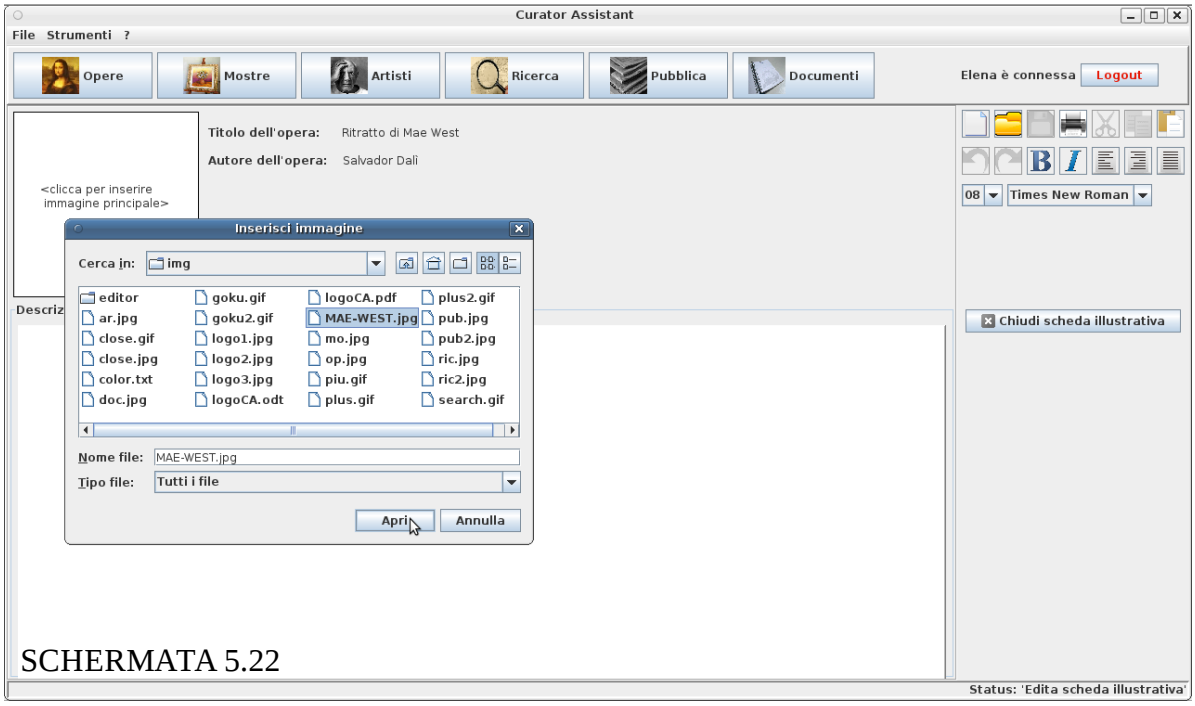
SCHERMATA 5.20

Elena decide di provare a inserire l'immagine principale della scheda, quindi clicca sullo spazio apposito e gli viene presentata una finestra dove è possibile selezionare file dal *filesystem* del proprio computer.



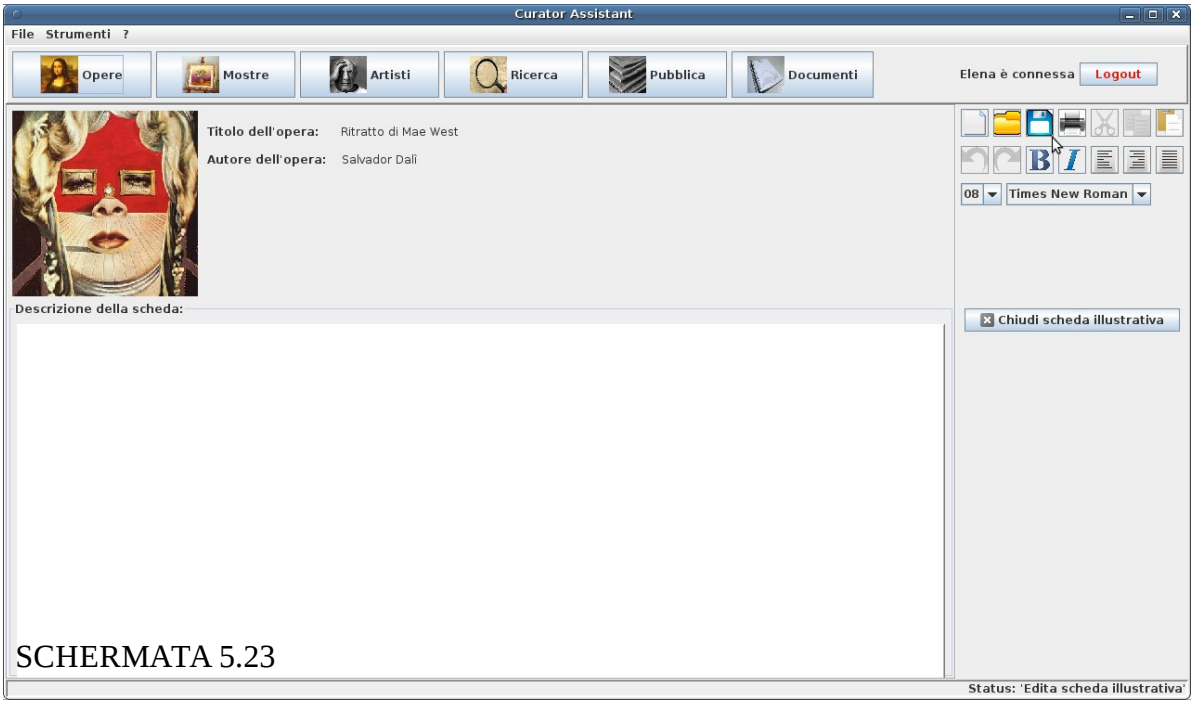
SCHERMATA 5.21

Elena seleziona l'immagine desiderata e clicca su 'Apri':



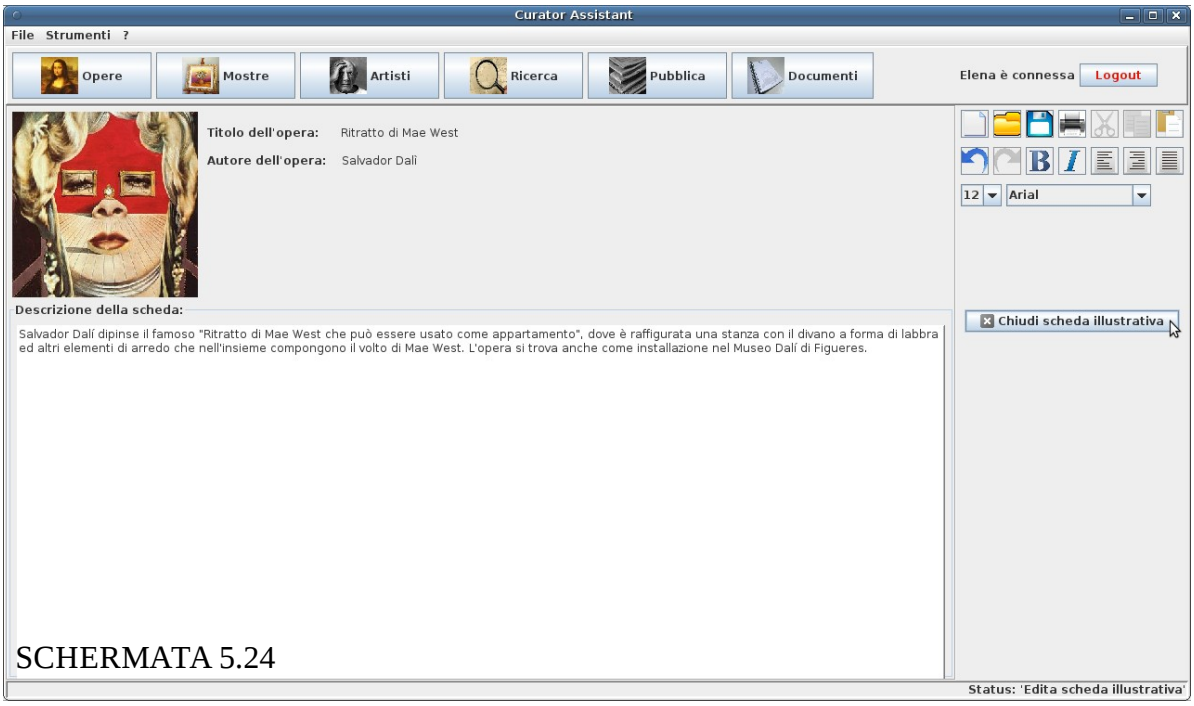
SCHERMATA 5.22

L'immagine è presentata nel riquadro apposito e il tasto 'salva' si abilita ora che è stata apportata una modifica alla scheda precedentemente vuota:



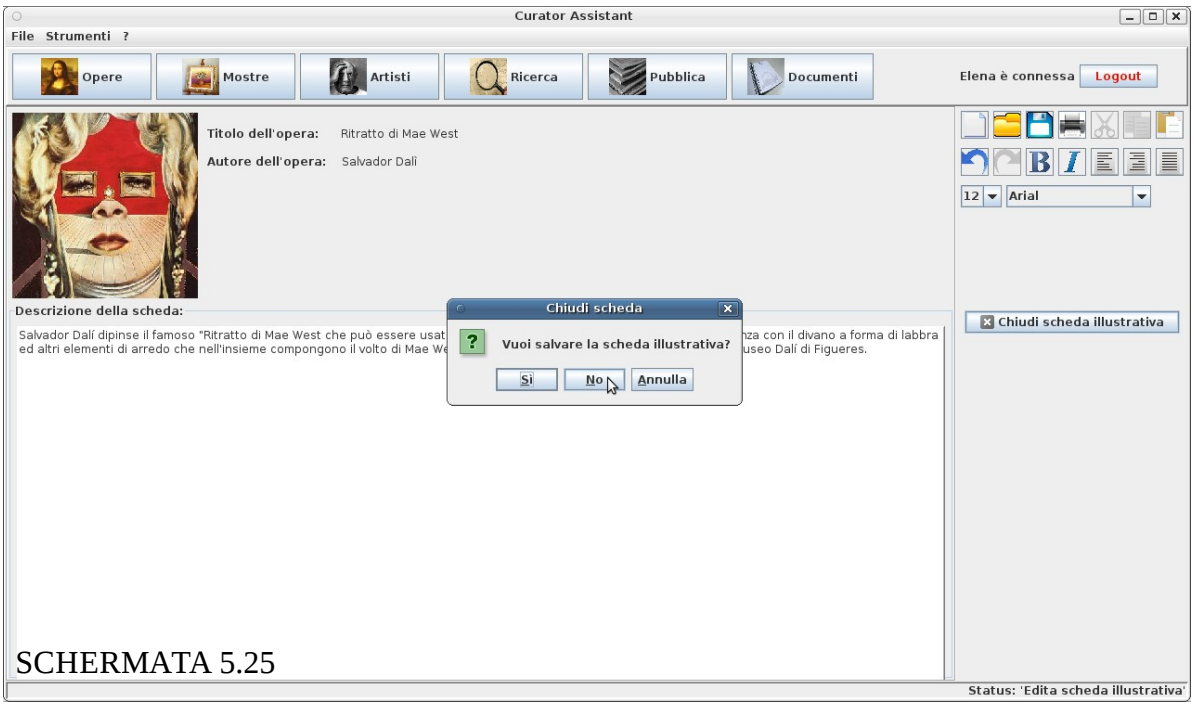
SCHERMATA 5.23

Elena prova anche ad abbozzare una descrizione dell'opera, poi decide di chiudere la scheda illustrativa:



SCHERMATA 5.24

Poiché sono state apportate modifiche alla scheda illustrativa vuota il sistema chiede se si vuole salvare la scheda. Elena chiude senza salvare:



SCHERMATA 5.25

5.4.5 Valutazione “Catalogare nuova opera”

Visibilità dello stato del sistema:

L'utente è sempre avvisato sui cambiamenti di stato del sistema scaturiti dalla sua interazione, ad esempio nella schermata 5.14 ciò che l'utente digita nel campo 'Autore' aggiorna dinamicamente la lista degli artisti suggeriti e se l'utente continua la compilazione con il nome di un'artista non valido, tale stato è comunicato all'utente stesso come si può vedere nella schermata 5.16. Nella schermata 5.23 il pulsante 'salva' si abilita dopo che l'utente ha modificato la scheda illustrativa vuota, anche questa è una comunicazione dello stato del sistema all'utente che indica a quest'ultimo che è stato inserito qualcosa di nuovo non ancora salvato. La barra di stato aiuta ancora una volta l'utente a capire cosa sta facendo, dalla schermata 5.20 in poi tale barra ricorda all'utente che sta nella fase di *editing* di una scheda illustrativa.

Corrispondenza fra il mondo reale e il sistema:

La progettazione dell'interfaccia è sempre focalizzata sul compito, l'utente (per noi l'utente tipo è Elena) trova nell'interfaccia i passi che venivano seguiti prima dell'istallazione del sistema nell'esecuzione del compito. Il linguaggio dell'interfaccia è vicino all'utente tipico, termini come 'Collocazione' sono presi dal gergo museale.

Libertà e controllo da parte degli utenti:

Ancora una volta la finestra 'Inserisci opera' da libertà all'utente di inserire i dati nell'ordine che preferisce (schermata 5.13), di decidere se e quando inserire i dati opzionali e permette di annullare l'operazione di inserimento. Per quanto riguarda invece l'*editor* di scheda illustrativa (schermata 5.20 – 5.25) esso fornisce le classiche funzioni di controllo di *undo* e *redo* per uscire da uno stato indesiderato nel caso l'utente abbia selezionato funzioni dell'*editor* per errore.

Il menu di pulsanti del “Contenitore principale” (Opere, Mostre ecc.) è mostrato anche durante la fase di *editing* scheda in modo che l'utente non perda il controllo sull'intero sistema.

Consistenza e standard:

L'*editor* di scheda illustrativa possiede un pannello di controllo con pulsanti tipici di un qualsiasi *editor* (salva, stampa, taglia, *undo*, *redo* ecc.). Le icone di tali pulsanti sono evocative e seguono le convenzioni comuni in modo che l'utente possa riconoscerli attraverso la "legge dell'esperienza passata".

Prevenzione degli errori:

È stata di nuovo data particolare enfasi alla prevenzione dell'errore; la finestra 'Inserisci opera' (Schermata 5.13 – 5.18) non permette la possibilità di confermare se i dati inseriti non sono corretti e in particolare il campo 'Autore' fornisce suggerimenti immediati (Schermata 5.14) per inserire nomi di artisti validi cercando di evitare che l'utente inserisca nomi non presenti nel sistema. In caso l'utente inserisca ugualmente un nome di artista invalido ignorando i suggerimenti allora ciò è subito comunicato all'utente (Schermata 5.16). Si evita quindi che l'utente inserisca dati che solo successivamente scopre essere scorretti perché questo creerebbe false aspettative nell'utente. Ancora una volta le *checkbox* sono state sostituite, dove possibile, a campi liberi per far si che l'utente scelga il valore del campo in un insieme prestabilito di valori corretti, l'uso di una *checkbox* per il campo 'Autore' non era ammissibile perché il numero di artisti nel sistema può diventare molto grande. Nelle schermate relative all'*editor* di scheda si può vedere invece che alcuni pulsanti del pannello di controllo, come salva ed *undo*, sono inizialmente disabilitati per evitare interazioni scorrette (non avrebbe senso salvare o annullare l'ultima azione se l'utente non ha ancora apportato alcuna modifica alla scheda).

Riconoscere piuttosto che ricordare:

Sempre relativamente alle schermate dell'*editor* l'utente riconosce le funzioni associate ai pulsanti del pannello di controllo (salva, stampa, ecc.) perché i pulsanti stessi sono visibili e hanno un aspetto familiare. L'utente non deve quindi ricordare come svolgere le operazioni di *editing* più comuni. Anche la frase <clicca qui per inserire immagine principale> (Schermata 5.20) nell'apposito riquadro segue la filosofia "riconoscere più che ricordare".

Flessibilità ed efficienza d'uso:

Come accadeva per la finestra 'Inserisci Mostra' anche la finestra 'Inserisci Opera' ha una certa flessibilità perché all'inizio è presentata in complessità ridotta e può aumentare la sua complessità a seconda se l'utente sceglie di inserire i dati opzionali. Una impostazione simile potrebbe essere adottata per l'editor di scheda illustrativa, la menubar (file, strumenti) sempre presente nella varie schermate potrebbe modificarsi quando si entra nella fase di modifica scheda illustrativa e aggiungere voci di menu dedicate ad un editing sofisticato riservato gli utenti esperti (non implementato).

Design minimalista:

L'editor di scheda presenta soltanto gli oggetti di interazione più utilizzati e più essenziali. Come detto prima gli oggetti di interazione avanzati potrebbero essere nascosti nella *menubar*.

Aiutare gli utenti a riconoscere gli errori, diagnosticarli e correggerli:

Il messaggio di errore della 'Schermata 5.16' è espresso in linguaggio semplice e comprensibile. La lista di artisti che compare mentre l'utente modifica il campo 'Autore' serve proprio a scongiurare eventuali errori di digitazione cosicché l'utente possa selezionare l'artista giusto invece di completarne il nome. Se invece intenzione dell'utente è proprio inserire il nome di un'artista nuovo che ancora non è presente nel sistema una soluzione per correggere la situazione di errore (come da specifiche non potrebbe esserci un'opera senza artista valido associato) potrebbe essere aggiungere un pulsante 'Inserisci artista' di fianco al messaggio di errore in modo che l'utente possa inserire un'artista in concomitanza con l'inserimento di un'opera senza interrompere il dialogo (non implementato).

Guida e documentazione:

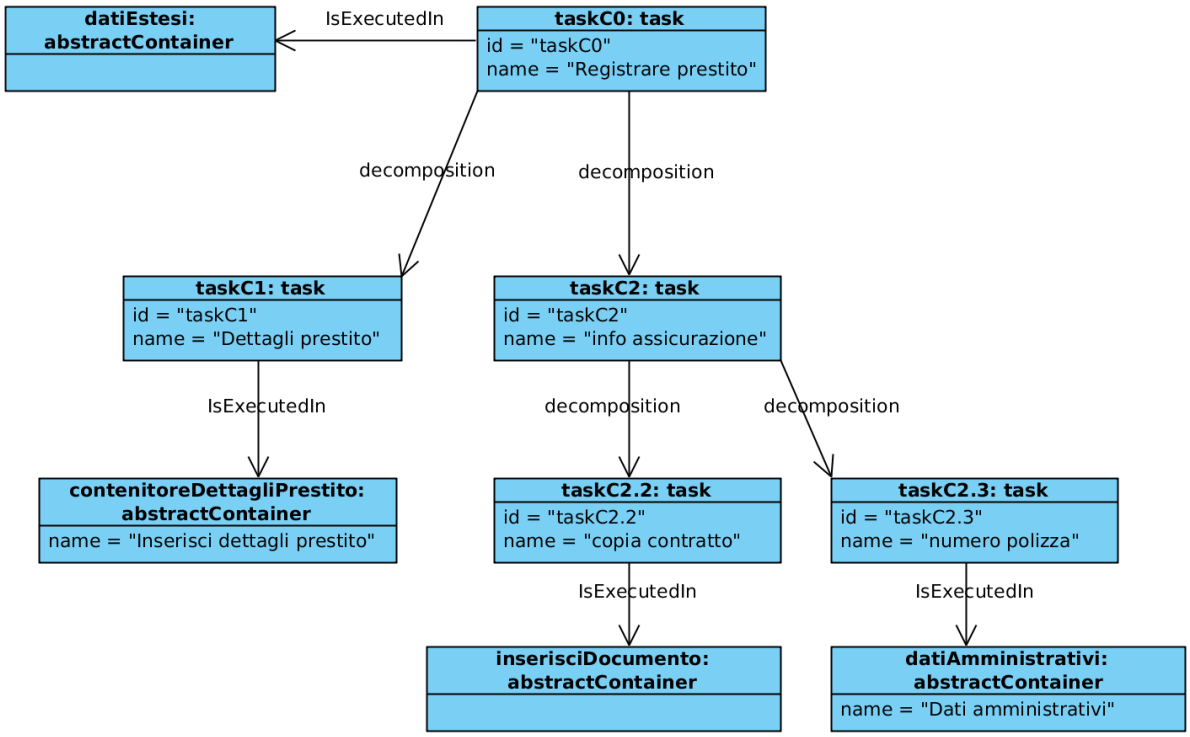
Dal momento che l'editor di scheda è presentato anche'esso nel "Contenitore principale" è ancora facilmente raggiungibile la voce di menu *help* della *menubar*.

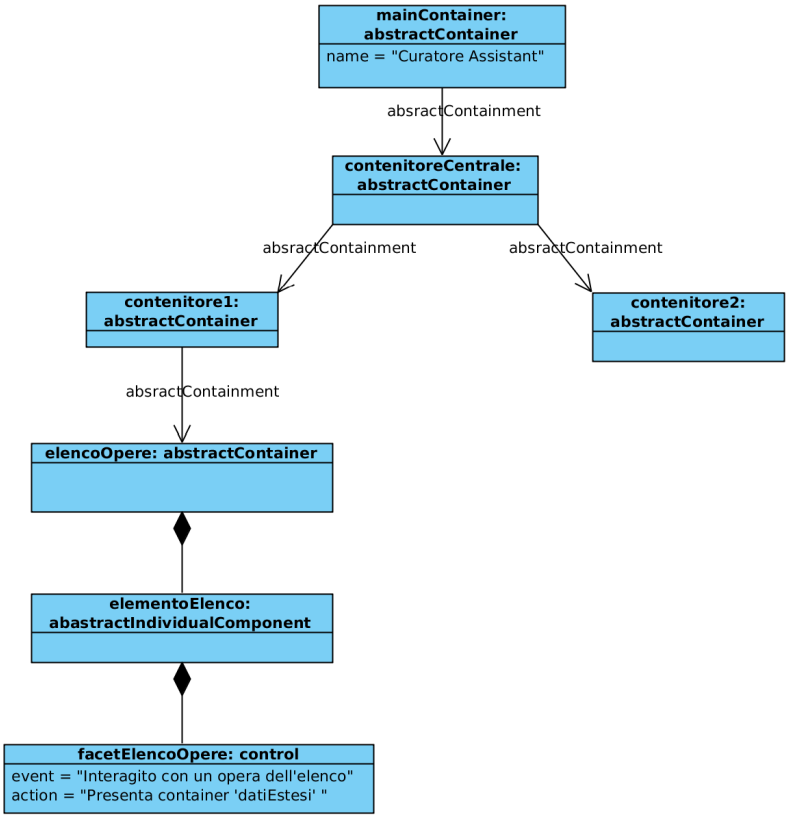
Altre considerazioni:

Poiché in questa parte di interfaccia è stato usato come modello di utente Elena, come specificato nel suo profilo essa soffre di problemi di emicrania. Una lettura delle schermate del sistema resa faticosa potrebbe aggravare il suo problema. A tal scopo il *font* dei testi delle schermate grafiche è stato scelto senza grazie e la dimensione del *font* non è quasi mai inferiore al corpo 12 per aumentare la *legibility* del testo stesso.

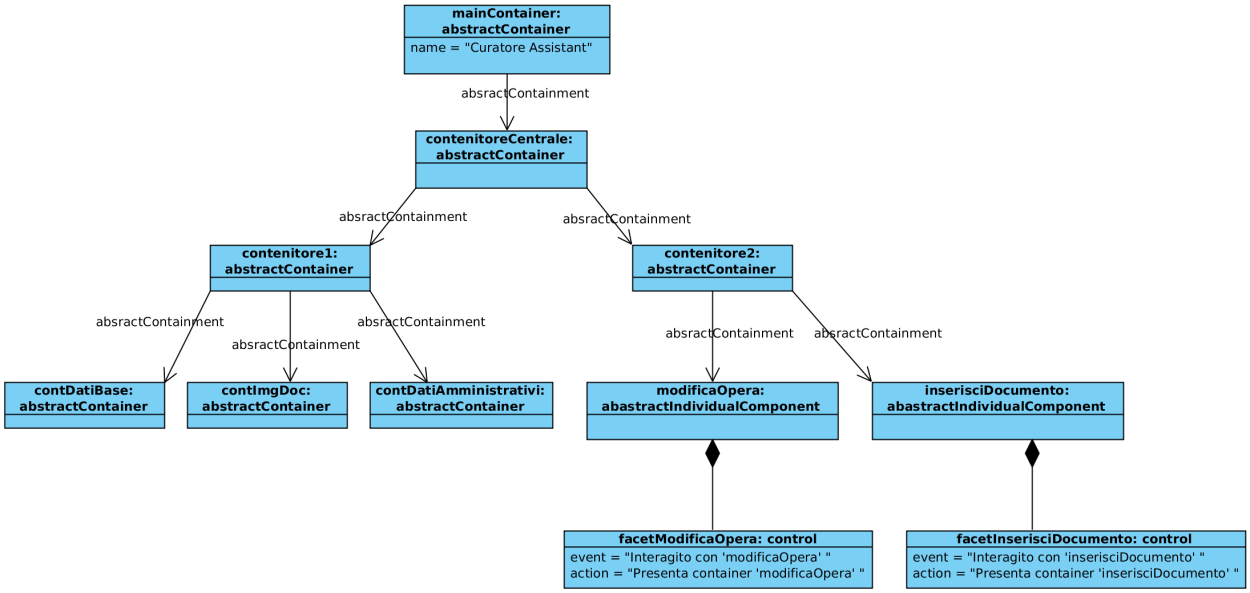
5.5 Realizzazione compito C

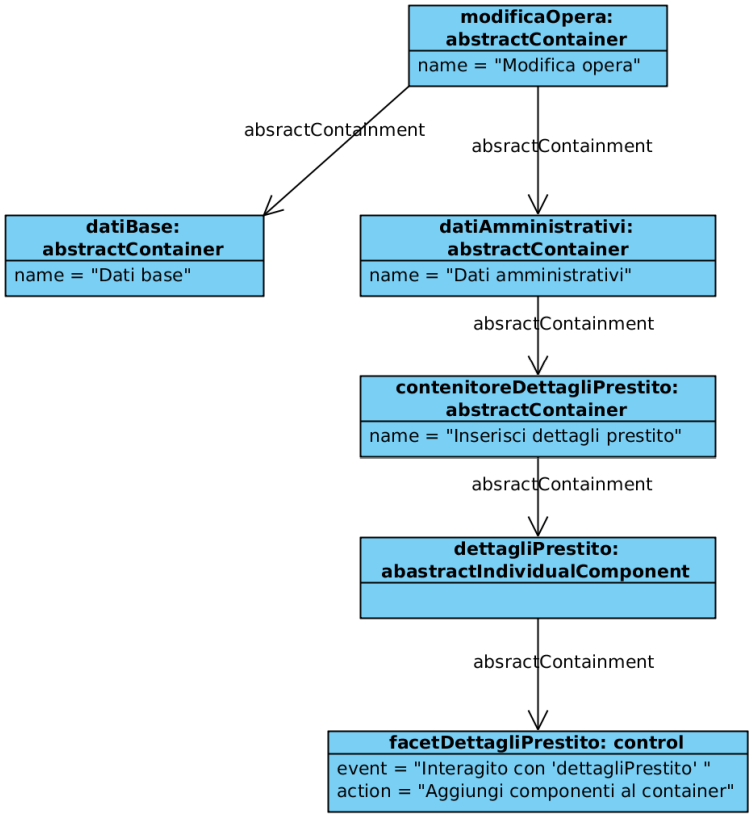
5.5.1 Progettazione astratta "Registrare prestito"



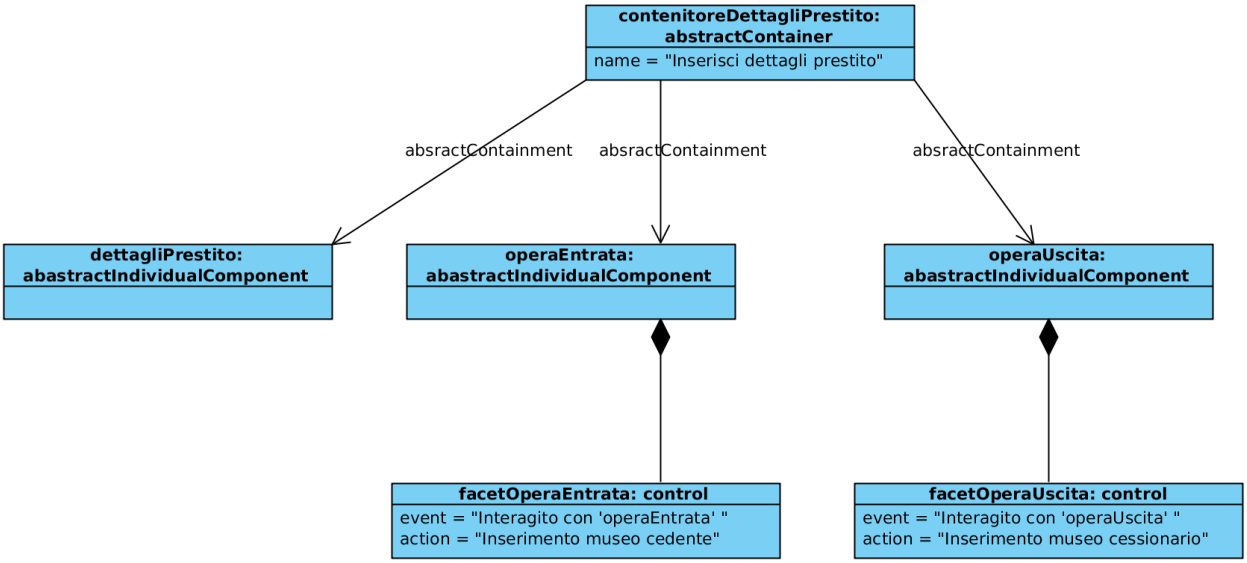


Se si interagisce con un'opera dell'elenco il 'contenitore1' e il 'contenitore2' del 'Contenitore principale' modificano i componenti che contengono presentando i “dati estesi” dell'opera con cui si è interagito:

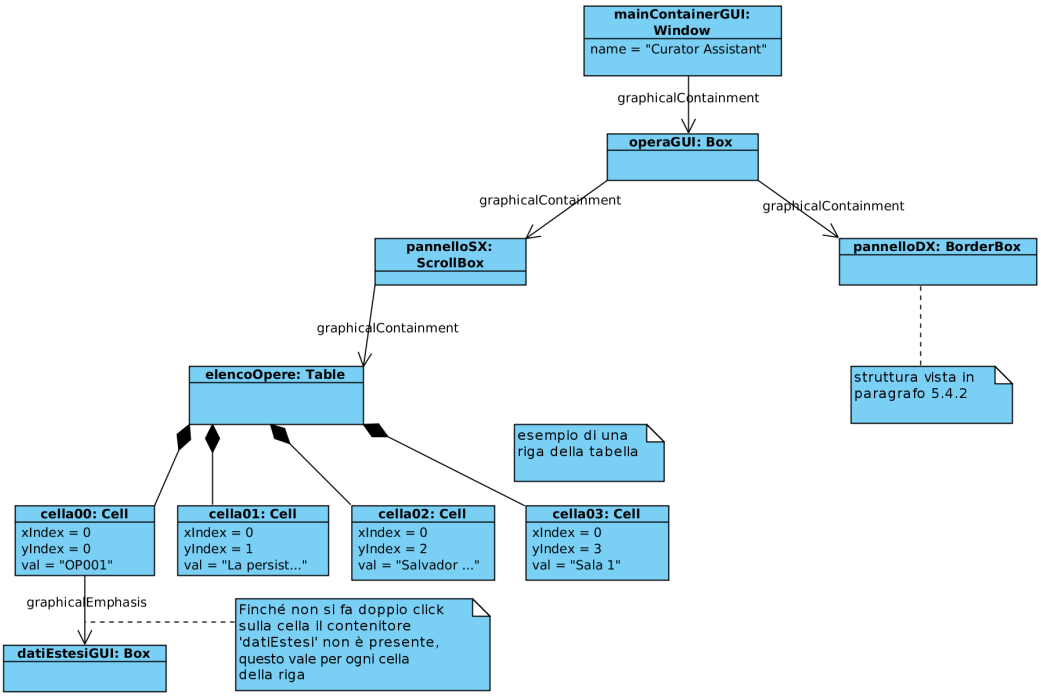




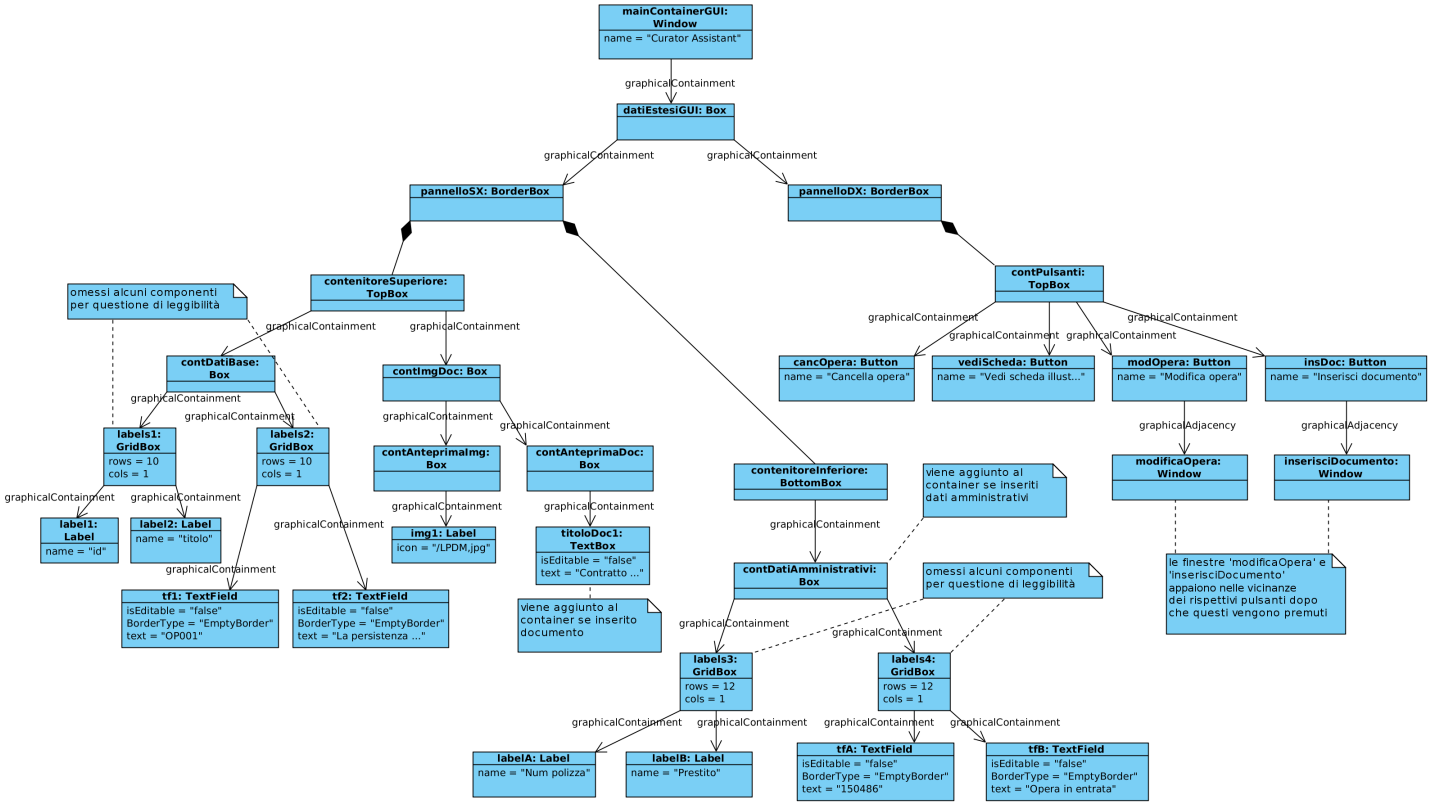
Interagendo con il componente individuale astratto 'dettagliPrestito' si aggiungono nuovi componenti al contenitore che permettono l'aggiunta dei dettagli sull'eventuale prestito dell'opera:



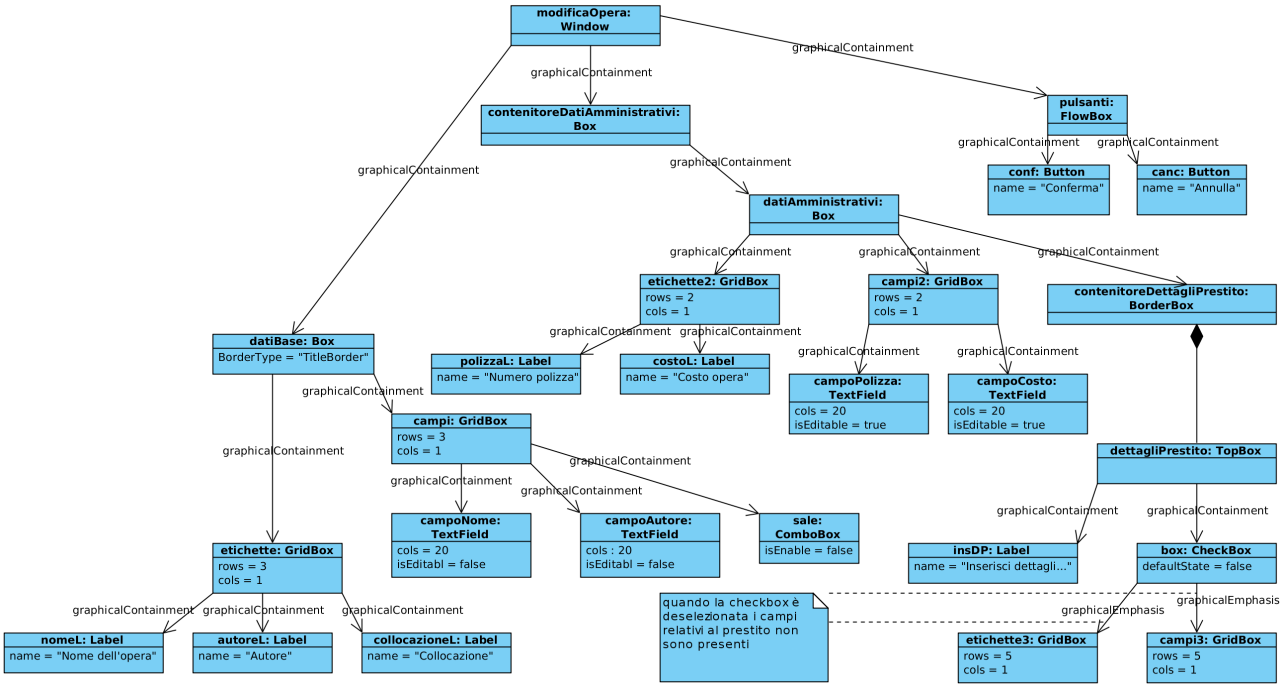
5.5.2 Progettazione concreta “Registrare prestito”



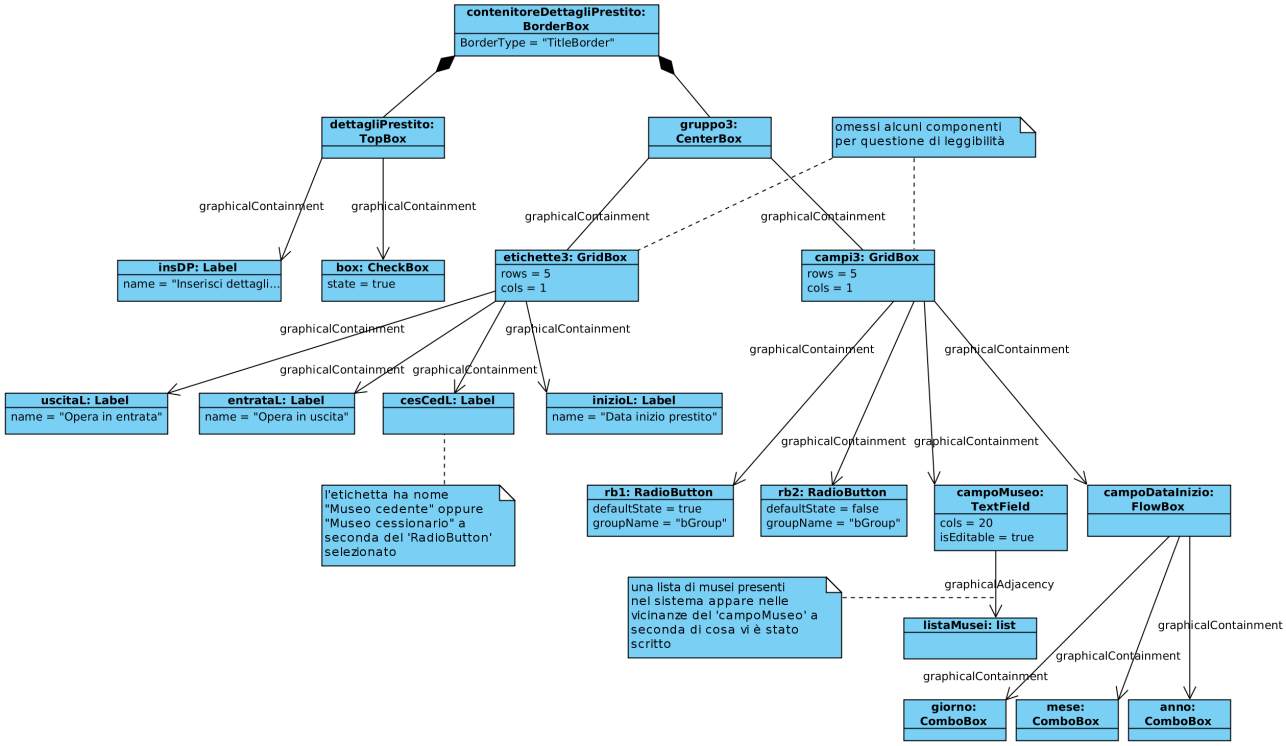
Come nella progettazione astratta se si interagisce con un'opera dell'elenco il 'Contenitore principale' modifica i suoi oggetti di contenimento mostrando i “dati estesi” dell'opera:



Vediamo in dettaglio la struttura concreta della finestra 'modificaOpera':



Vediamo quali sono i componenti che si aggiungono alla finestra selezionando la *checkbox* 'Inserisci dettagli prestito' :



5.5.3 Implementazione fisica “Registrare prestito”

Completare il 'Frammento omesso 1' del codice 5.1.3 con il seguente frammento, oltre a quanto già aggiunto nei paragrafi precedenti:

```
final DatiEstesiGUI panDatiEstesi;
```

Completare il 'Frammento omesso 2' del codice 5.1.3 con il seguente frammento, oltre a quanto già aggiunto nei paragrafi precedenti:

```
panDatiEstesi = new DatiEstesiGUI(this);
```

Completare il 'Frammento omesso 5' del codice 5.4.3 con il seguente frammento:

```
if(s=="Dati estesi") panDatiEstesi.ridisegna();
```

Completare il 'Frammento omesso 6' del codice 5.4.3 con il seguente frammento:

```
tab.addMouseListener(new MyMouseListener2(menuPrincipale));
```

Aggiungere in coda alla classe 'OperaGUI' le seguente classe, oltre a quelle già aggiunte nei paragrafi precedenti:

```
class MyMouseListener2 implements MouseListener{
        MainContainerGUI menuPrincipale;
        public MyMouseListener2 (MainContainerGUI mP) { menuPrincipale=mP;}
        public void mouseClicked(MouseEvent e) {
                if(e.getClickCount()==2){
                        JTable tab = (JTable)e.getSource();
                        int riga = tab.getSelectedRow();
                        if(riga==0){ menuPrincipale.visualizza("Dati estesi", null); }
                }
        }
        public void mouseEntered(MouseEvent arg0) {}
        public void mouseExited(MouseEvent arg0) {}
        public void mousePressed(MouseEvent arg0) {}
        public void mouseReleased(MouseEvent arg0) {}
}
```

Aggiungere al package la classe 'DatiEstesiGUI':

```
package curatorAssistantGUI_betaVs;
import javax.swing.*;
import java.awt.*;
import javax.swing.text.*;
import javax.swing.tree.*;
import java.awt.Toolkit;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.FocusEvent;
import java.awt.event.FocusListener;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.io.File;

public class DatiEstesiGUI {

         JPanel panSX, panDX;
         JPanel contImgDoc;
         JPanel contAnteprimaImg;
         JPanel contInferiore=new JPanel(new BorderLayout());
         MainContainerGUI menuPrincipale;

         public DatiEstesiGUI(MainContainerGUI fin){

                  panSX = new JPanel(new BorderLayout());
                  panDX = new JPanel(new BorderLayout());
                  menuPrincipale=fin;

                  JPanel contVuoto = new JPanel();
                  contVuoto.setPreferredSize(new Dimension(1000,1000));

                  contAnteprimaImg = new JPanel(new FlowLayout(FlowLayout.LEFT));
                  JPanel contSuperiore = new JPanel(new BorderLayout());

                  JPanel contDatiBase = new JPanel(new FlowLayout(FlowLayout.LEFT));
                  JTextPane imgPrincipale = new JTextPane();
                  imgPrincipale.setPreferredSize(new Dimension(100,100));
                  imgPrincipale.setBorder(BorderFactory.createLineBorder(new Color(0,0,0)));
                  imgPrincipale.setEditable(false);

                  JPanel labels1 = new JPanel (new GridLayout(10,1));
                 labels1.add(new JLabel(""));
                  labels1.add(new JLabel("Identificativo:"));
                  labels1.add(new JLabel(""));
                  labels1.add(new JLabel("Titolo dell'opera:"));
                  labels1.add(new JLabel(""));
                  labels1.add(new JLabel("Autore dell'opera:"+" "+" "+" "+" "+" "+" "+" "+" "));
                  labels1.add(new JLabel(""));
                  labels1.add(new JLabel("Collocazione:"));
                  labels1.add(new JLabel(""));
                  labels1.add(new JLabel("Tipologia:"));

                  JPanel labels2 = new JPanel (new GridLayout(10,1));
                  labels2.add(new JLabel(""));
                  JTextField tf0=new JTextField("OP001");
                  tf0.setEditable(false);
                  tf0.setBorder(BorderFactory.createEmptyBorder());
                  labels2.add(tf0);
                  labels2.add(new JLabel(""));

                  JTextField tf1=new JTextField("La persistenza della memoria");
                  tf1.setEditable(false);
                  tf1.setBorder(BorderFactory.createEmptyBorder());
                  labels2.add(tf1);
                  labels2.add(new JLabel(""));

                  JTextField tf2=new JTextField("Salvador Dalì");
                  tf2.setEditable(false);
                  tf2.setBorder(BorderFactory.createEmptyBorder());
                  labels2.add(tf2);
                  labels2.add(new JLabel(""));
```

```
                JTextField tf3=new JTextField("Sala 1");
                tf3.setEditable(false);
                tf3.setBorder(BorderFactory.createEmptyBorder());
                labels2.add(tf3);
                labels2.add(new JLabel(""));

                JTextField tf4=new JTextField("Quadro");
                tf4.setEditable(false);
                tf4.setBorder(BorderFactory.createEmptyBorder());
                labels2.add(tf4);

                Box group = Box.createHorizontalBox();
                group.add(labels1);
                group.add(labels2);

                contDatiBase.add(group);

                Image i = Toolkit.getDefaultToolkit().createImage(
                "/home/macash/Desktop/CollegamentoIUM/img/LPDM.jpg");
                i=i.getScaledInstance(150,150,Image.SCALE_DEFAULT);
                ImageIcon ic = new ImageIcon(i);

                JLabel img1 = new JLabel();
                img1.setIcon(ic);
                img1.setBorder(BorderFactory.createLineBorder(new Color(0,0,0)));

                contAnteprimaImg.add(new JLabel(" "));
                contAnteprimaImg.add(img1);

                contImgDoc = new JPanel(new BorderLayout());
                contImgDoc.add(contAnteprimaImg, BorderLayout.CENTER);

                contSuperiore.add(contDatiBase, BorderLayout.NORTH);
                contSuperiore.add(new JLabel("  "), BorderLayout.CENTER);
                contSuperiore.add(contImgDoc, BorderLayout.SOUTH);

                contDatiBase.setBorder(BorderFactory.createTitledBorder("DATI BASE"));
                contAnteprimaImg.setBorder(BorderFactory.createTitledBorder("IMMAGINI"));

                JPanel empty = new JPanel();
                empty.setPreferredSize(new Dimension(500,800));

                panSX.add(contSuperiore, BorderLayout.NORTH);
                panSX.add(contVuoto, BorderLayout.CENTER);
                panSX.setBorder(BorderFactory.createEtchedBorder());

                panDX.setPreferredSize(new Dimension(290,100));

                JPanel contPulsanti = new JPanel(new GridLayout(12,1));
                JPanel contModOpera = new JPanel();
                JPanel contCancOpera = new JPanel();
                JPanel contVediScheda = new JPanel();
                JPanel contInsDoc = new JPanel();
JButton modOpera = new JButton(" "+" "+" "+" "+" "+" "+" "+" "+"Modifica opera"+" "+" "+" "+" "+" "+" "+" "+" ");
String s = new String(" "+" "+" "+" "+" "+" "+" "+" "+"Cancella opera"+" "+" "+" "+" "+" "+" "+" "+" ");
                JButton cancOpera = new JButton(s);
                JButton vediScheda = new JButton(" "+"Vedi scheda illustrativa");
                JButton insDoc = new JButton(" "+" "+" "+"Inserisci documento"+" "+" "+" "+" ");

                insDoc.addActionListener(new ActionListener(){


                        public void actionPerformed(ActionEvent e) {

                                        JComponent c = (JComponent)e.getSource();

                                        JFileChooser inserisciDocumento = new JFileChooser();
                                        inserisciDocumento.setDialogTitle("Inserisci documento");
                                        int sel = inserisciDocumento.showOpenDialog(c);

                                        if(sel==JFileChooser.APPROVE_OPTION) {

                                        AggiungiDoc();

                                        }

                        }

            });
```

```
            modOpera.addActionListener(new MyModOperaActionListener(menuPrincipale, this));

            Image i2 = Toolkit.getDefaultToolkit().createImage(
            "/home/macash/Desktop/CollegamentoIUM/img/edit2.gif");
            i2=i2.getScaledInstance(15,15,Image.SCALE_DEFAULT);
            ImageIcon ic2 = new ImageIcon(i2);

            modOpera.setIcon(ic2);

            Image i3 = Toolkit.getDefaultToolkit().createImage(
            "/home/macash/Desktop/CollegamentoIUM/img/remove.gif");
            i3=i3.getScaledInstance(15,15,Image.SCALE_DEFAULT);
            ImageIcon ic3 = new ImageIcon(i3);

            cancOpera.setIcon(ic3);

            Image i4 = Toolkit.getDefaultToolkit().createImage(
            "/home/macash/Desktop/CollegamentoIUM/img/eye.gif");
            i4=i4.getScaledInstance(16,16,Image.SCALE_DEFAULT);
            ImageIcon ic4 = new ImageIcon(i4);

            vediScheda.setIcon(ic4);

            Image i5 = Toolkit.getDefaultToolkit().createImage(
            "/home/macash/Desktop/CollegamentoIUM/img/plus2.gif");
            i5=i5.getScaledInstance(14,14,Image.SCALE_DEFAULT);
            ImageIcon ic5 = new ImageIcon(i5);

            insDoc.setIcon(ic5);

            contModOpera.add(modOpera);
            contCancOpera.add(cancOpera);
            contVediScheda.add(vediScheda);
            contInsDoc.add(insDoc);

            contPulsanti.add(new JLabel(""));
            contPulsanti.add(contModOpera);
            contPulsanti.add(contCancOpera);
            contPulsanti.add(contVediScheda);
            contPulsanti.add(contInsDoc);

            JPanel empty2 = new JPanel();
            empty2.setPreferredSize(new Dimension(300,500));

            panDX.add(contPulsanti, BorderLayout.NORTH);
            panDX.add(empty2, BorderLayout.SOUTH);

    }

    public void ridisegna(){

            menuPrincipale.pannelloSX=panSX;
            menuPrincipale.pannelloDX=panDX;

    };

    public void aggiungiModifiche(String[] ss, boolean prest){

            JPanel contdatiAmministrativi = new JPanel(new FlowLayout(FlowLayout.LEFT));
            JPanel labels3 = new JPanel (new GridLayout(12,1));
            JPanel labels4 = new JPanel (new GridLayout(12,1));

            if(prest==false) {

                    labels3.add(new JLabel(""));
                    labels3.add(new JLabel("Numero polizza:"));
                    labels3.add(new JLabel(""));
                    labels3.add(new JLabel(""));
                    labels3.add(new JLabel(""));
                    labels3.add(new JLabel(""));
                    labels3.add(new JLabel(""));
                    labels3.add(new JLabel(""));
                    labels3.add(new JLabel(""));
                    labels3.add(new JLabel(""));
                    labels3.add(new JLabel(""));
                    labels3.add(new JLabel(""));
                    labels4.add(new JLabel(""));
                    JTextField tf=new JTextField();
                    tf.setText(" "+" "+" "+" "+ss[0]);
```

```
                tf.setEditable(false);
                tf.setBorder(BorderFactory.createEmptyBorder());
                labels4.add(tf);
                labels4.add(new JLabel(""));
                labels4.add(new JLabel(""));
                labels4.add(new JLabel(""));
                labels4.add(new JLabel(""));
                labels4.add(new JLabel(""));
                labels4.add(new JLabel(""));
                labels4.add(new JLabel(""));
                labels4.add(new JLabel(""));
                labels4.add(new JLabel(""));
                labels4.add(new JLabel(""));

        }
        else{

                labels3.add(new JLabel(""));
                labels3.add(new JLabel("Numero polizza:"));
                labels3.add(new JLabel(""));
                labels3.add(new JLabel("Prestito:"));
                labels3.add(new JLabel(""));
                labels3.add(new JLabel(ss[2]));
                labels3.add(new JLabel(""));
                labels3.add(new JLabel("Data inizio prestito:"+" "+" "+" "+" "));
                labels3.add(new JLabel(""));
                labels3.add(new JLabel("Data fine prestito:"));
                labels3.add(new JLabel(""));
                labels3.add(new JLabel(""));
                labels4.add(new JLabel(""));
                JTextField tf=new JTextField();
                tf.setText(""+ss[0]);
                tf.setEditable(false);
                tf.setBorder(BorderFactory.createEmptyBorder());
                labels4.add(tf);

                labels4.add(new JLabel(""));

                JTextField tfA=new JTextField();
                tfA.setText(""+ss[1]);
                tfA.setEditable(false);
                tfA.setBorder(BorderFactory.createEmptyBorder());
                labels4.add(tfA);

                labels4.add(new JLabel(""));

                JTextField tfB=new JTextField();
                tfB.setText("MoMA di New York");
                tfB.setEditable(false);
                tfB.setBorder(BorderFactory.createEmptyBorder());
                labels4.add(tfB);

                labels4.add(new JLabel(""));

                JTextField tfC=new JTextField();
                tfC.setText(""+ss[3]);
                tfC.setEditable(false);
                tfC.setBorder(BorderFactory.createEmptyBorder());
                labels4.add(tfC);

                labels4.add(new JLabel(""));

                JTextField tfD=new JTextField();
                tfD.setText(""+ss[4]);
                tfD.setEditable(false);
                tfD.setBorder(BorderFactory.createEmptyBorder());
                labels4.add(tfD);

                labels4.add(new JLabel(""));
                labels4.add(new JLabel(""));


        }

        Box grp = Box.createHorizontalBox();
        grp.add(labels3);
        grp.add(labels4);

        contdatiAmministrativi.add(grp);
        contdatiAmministrativi.setBorder(BorderFactory.createTitledBorder("DATI AMMINISTRATIVI"));
```

```
                contInferiore.add(contdatiAmministrativi, BorderLayout.NORTH);
                panSX.add(contInferiore, BorderLayout.SOUTH);

                menuPrincipale.visualizza("Dati estesi", null);

        }


        public void AggiungiDoc(){

                JTextPane titoloDoc1 = new JTextPane();
                titoloDoc1.setEditable(false);

                SimpleAttributeSet sott = new SimpleAttributeSet();
                StyleConstants.setUnderline(sott, true);
                titoloDoc1.setCharacterAttributes(sott, true);
                titoloDoc1.setText("Contratto assicurativo.pdf");
                titoloDoc1.setForeground(new Color(0,0,255));

                JPanel contAnteprimaDoc = new JPanel(new FlowLayout(FlowLayout.LEFT));
                titoloDoc1.setBackground(contAnteprimaDoc.getBackground());
                contAnteprimaDoc.add(titoloDoc1);

                contAnteprimaDoc.setBorder(BorderFactory.createTitledBorder("DOCUMENTI"));

                contImgDoc.remove(contAnteprimaImg);

                contAnteprimaDoc.setPreferredSize(new Dimension(500,contAnteprimaImg.getHeight()));

                contImgDoc.add(contAnteprimaImg, BorderLayout.CENTER);
                contImgDoc.add(contAnteprimaDoc, BorderLayout.EAST);

                menuPrincipale.visualizza("Dati estesi", null);

        }

}
```

Aggiungere in coda alla classe 'DatiEstesiGUI' le seguenti classi:

```
class MyModOperaActionListener implements ActionListener{

        MainContainerGUI menuPrincipale;
        DatiEstesiGUI deGUI;
        MyItemListener2 mil2;

        JButton conf, canc;
        JFrame modificaOpera;
        JTextField campoNome = new JTextField(20);
        JTextField campoAutore = new JTextField(20);
        JTextField campoPolizza = new JTextField(20);
        JTextField campoCosto = new JTextField(20);

        Box gruppo = Box.createHorizontalBox();
        Box gruppo2 = Box.createHorizontalBox();

        GridLayout gl =new GridLayout(5,1);
        GridLayout gl2 =new GridLayout(3,1);
        JPanel etichette = new JPanel(gl);
        JLabel nomeL = new JLabel(" Nome dell'opera:"+" "+" "+" "+" "+" "+" ");
        JLabel autoreL = new JLabel(" Autore:");
        JLabel collocazioneL = new JLabel(" Collocazione:");
        JLabel tipologiaL = new JLabel(" Tipologia:");
        JPanel campi = new JPanel(gl);
        JPanel text1 = new JPanel();
        JPanel text2 = new JPanel();

        DefaultComboBoxModel model1 = new DefaultComboBoxModel();
        JComboBox tipo = new JComboBox(model1);
        JPanel tipologia = new JPanel(new FlowLayout(FlowLayout.LEFT));

        DefaultComboBoxModel model1_2 = new DefaultComboBoxModel();
        JComboBox sale = new JComboBox(model1_2);
        JPanel collocazione = new JPanel(new FlowLayout(FlowLayout.LEFT));
```

```
JPanel etichette2 = new JPanel(gl2);
JLabel polizzaL = new JLabel(" Numero polizza:"+" "+" "+" "+" "+" "+" ");
JLabel costoL = new JLabel(" Costo opera:");

JPanel campi2 = new JPanel(gl2);
JPanel text3 = new JPanel();
JPanel text4 = new JPanel();

JCheckBox box;

boolean sug=false;
boolean b=false;
boolean statoConf1 =true, statoConf2 = true;


public MyModOperaActionListener(MainContainerGUI mP , DatiEstesiGUI deG){deGUI=deG; menuPrincipale=mP;}

public void actionPerformed(ActionEvent e){

        modificaOpera = new JFrame("Modifica opera");

        text1.add(campoNome);
        text2.add(campoAutore);

        campoNome.setText("La persistenza della memoria");
        campoNome.setEditable(false);
        campoNome.setBorder(BorderFactory.createLineBorder(new Color(0,0,0)));

        campoAutore.setText("Dalì Salvador");
        campoAutore.setEditable(false);
        campoAutore.setBorder(BorderFactory.createLineBorder(new Color(0,0,0)));

        model1.addElement("Quadro");
        model1.addElement("Scultura");
        model1.addElement("Altro");

        tipologia.add(tipo);
        tipo.setEnabled(false);

        model1_2.addElement("Sala1");
        model1_2.addElement("Sala2");
        model1_2.addElement("Sala3");
        model1_2.addElement("Sala4");
        model1_2.addElement("Sala5");
        model1_2.addElement("Sala6");

        collocazione.add(sale);
        sale.setEnabled(false);

        etichette.add(nomeL);
        etichette.add(autoreL);
        etichette.add(collocazioneL);
        etichette.add(tipologiaL);

        campi.add(text1);
        campi.add(text2);
        campi.add(collocazione);
        campi.add(tipologia);

        gruppo.add(etichette);
        gruppo.add(campi);
        gruppo.setBorder(BorderFactory.createTitledBorder("DATI BASE"));

        JPanel datiBase = new JPanel(new FlowLayout(FlowLayout.LEFT));

        datiBase.add(gruppo);

        JPanel datiAmministrativi = new JPanel(new BorderLayout());

        datiAmministrativi.setBorder(BorderFactory.createTitledBorder("DATI AMMINISTRATIVI"));

        text3.add(campoPolizza);
        text4.add(campoCosto);

        gruppo2.add(etichette2);
        gruppo2.add(campi2);

        etichette2.add(polizzaL);
        etichette2.add(costoL);
```

```
                campi2.add(text3);
                campi2.add(text4);

                JPanel contenitoreDettagliPrestito = new JPanel(new BorderLayout());
                JPanel dettagliPrestito = new JPanel(new FlowLayout(FlowLayout.LEFT));
                JLabel insDP = new JLabel("Inserisci dettagli prestito: "+" ");
                box = new JCheckBox();
                box.addItemListener(new MyItemListener2(contenitoreDettagliPrestito,modificaOpera,box,this));

                dettagliPrestito.add(insDP);
                dettagliPrestito.add(box);
                contenitoreDettagliPrestito.add(dettagliPrestito, BorderLayout.NORTH);

                datiAmministrativi.add(new JLabel(" "), BorderLayout.NORTH);
                datiAmministrativi.add(gruppo2, BorderLayout.CENTER);
                datiAmministrativi.add(contenitoreDettagliPrestito, BorderLayout.SOUTH);

                JPanel contenitoreDatiAmministrativi = new JPanel(new FlowLayout(FlowLayout.LEFT));
                contenitoreDatiAmministrativi.add(datiAmministrativi, BorderLayout.NORTH);

                JPanel pulsanti = new JPanel(new FlowLayout());
                canc = new JButton("Annulla");
                conf = new JButton("Conferma");

                pulsanti.add(canc);
                pulsanti.add(conf);

                campoPolizza.addKeyListener(new MyKeyListener3(campoPolizza, conf, this));
                conf.addActionListener(new MyConfActionListener3(deGUI, this));

                modificaOpera.add(datiBase,BorderLayout.NORTH);
                modificaOpera.add(contenitoreDatiAmministrativi,BorderLayout.CENTER);
                modificaOpera.add(pulsanti,BorderLayout.SOUTH);

                modificaOpera.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
                modificaOpera.pack();
                modificaOpera.setSize(420,600);
                modificaOpera.setVisible(true);

        }

        public void impostaStatoConf1(boolean state) { statoConf1=state;}

        public void impostaStatoConf2(boolean state) { statoConf2=state;}

        public boolean dammiStatoConf() {return statoConf1&&statoConf2; }

}
```

```
class MyKeyListener3 implements KeyListener {

        MyModOperaActionListener myModOp;
        JTextField tx;
        JButton bot;

        MyKeyListener3(JTextField t, JButton b, MyModOperaActionListener mmO){ tx=t; bot=b; myModOp=mmO;}

                        public void keyPressed(KeyEvent e){}
                        public void keyTyped(KeyEvent e){}
                        public void keyReleased(KeyEvent e) {
                                if(!tx.getText().isEmpty()) {
                                        try{
                                                Integer.valueOf(tx.getText()).intValue();
                                                myModOp.impostaStatoConf1(true);
                                                bot.setEnabled(myModOp.dammiStatoConf());
                                        }
                                        catch (Exception ex){
                                                myModOp.impostaStatoConf1(false);
                                                bot.setEnabled(myModOp.dammiStatoConf());
                                        }
                                }
                                else bot.setEnabled(true);
                        }

}
```

```
class MyItemListener2 implements ItemListener {

                Boolean flag=false;
                JPanel contDetPrest;
                JFrame fin;
                JCheckBox box;

                GridLayout gl3 = new GridLayout(5,1);

                JRadioButton rb1,rb2;
                JPanel dataInizio, dataFine;
                JComboBox giorno, mese, anno, giorno2, mese2, anno2;

                Box gruppo3 = Box.createHorizontalBox();
                JPanel etichette3 = new JPanel(gl3);
                JLabel entrataL = new JLabel(" Opera in entrata:");
                JLabel uscitaL = new JLabel(" Opera in uscita:");
                JLabel cesCedL = new JLabel(" Museo cessionario:");
                JLabel inizioL = new JLabel(" Data inizio prestito:"+" "+" "+" "+" "+" ");
                JLabel fineL = new JLabel(" Data fine prestito:");

                JPanel campi3 = new JPanel(gl3);
                JPanel text5 = new JPanel();
                JPanel text6 = new JPanel();
                JPanel text7 = new JPanel();
                JTextField campoMuseo = new JTextField(20);
                JTextField campoDataInizio = new JTextField(20);
                JTextField campoDataFine = new JTextField(20);
                Boolean grpEmpty=true;

                MyModOperaActionListener myModOp;
                MyKeyListener4 mkl4;

                public MyItemListener2(JPanel p, JFrame f, JCheckBox b, MyModOperaActionListener mmO)
                {contDetPrest=p; fin=f; box=b; myModOp=mmO;}

                public void itemStateChanged(ItemEvent e){

                        if(flag==null){}
                        else {
                                if(flag==false) flag=true;
                                else flag=false;

                                if(flag==true){

                                        myModOp.mil2=this;
                                        myModOp.impostaStatoConf2(false);
                                        myModOp.conf.setEnabled(myModOp.dammiStatoConf());

                                        fin.setSize(450,650);
                                        contDetPrest.setBorder(BorderFactory.createTitledBorder(""));

                                        if(grpEmpty==true){

                                                rb1 = new JRadioButton();
                                                rb2 = new JRadioButton();

                                                final ButtonGroup bGroup = new ButtonGroup();

                                                bGroup.add(rb1);
                                                bGroup.add(rb2);

                                                rb1.setSelected(true);

                                                etichette3.add(uscitaL);
                                                etichette3.add(entrataL);
                                                etichette3.add(cesCedL);
                                                etichette3.add(inizioL);
                                                etichette3.add(fineL);

                                                text5.add(campoMuseo);
                                                text6.add(campoDataInizio);
                                                text7.add(campoDataFine);

                                                mkl4 = new MyKeyListener4(myModOp, this);
                                                campoMuseo.addKeyListener(mkl4);

                                                campi3.add(rb1);
                                                campi3.add(rb2);
                                                campi3.add(text5);
```

```
                        rb1.addItemListener(new ItemListener(){


                                public void itemStateChanged(ItemEvent e) {
                                        JRadioButton rb= (JRadioButton)e.getSource();

                if(rb.isSelected()){ cesCedL.setText(" Museo cessionario:");}
                else {cesCedL.setText(" Museo cedente:");}

                                }

                        });

                        DefaultComboBoxModel model1 = new DefaultComboBoxModel();
                        model1.addElement("01");
                        model1.addElement("02");
                        model1.addElement("03");
                        model1.addElement("04");
                        model1.addElement("05");
                        model1.addElement("06");
                        giorno = new JComboBox(model1);

                        DefaultComboBoxModel model2 = new DefaultComboBoxModel();
                        model2.addElement("Settembre");
                        model2.addElement("Ottobre");
                        model2.addElement("Novembre");
                        model2.addElement("Dicembre");
                        mese = new JComboBox(model2);

                        DefaultComboBoxModel model3 = new DefaultComboBoxModel();
                        model3.addElement("2012");
                        model3.addElement("2013");
                        model3.addElement("2014");
                        anno = new JComboBox(model3);

                        dataInizio = new JPanel(new FlowLayout());
                        dataInizio.add(giorno);
                        dataInizio.add(mese);
                        dataInizio.add(anno);

                        campi3.add(dataInizio);

                        DefaultComboBoxModel model1_2 = new DefaultComboBoxModel();
                        model1_2.addElement("01");
                        model1_2.addElement("02");
                        model1_2.addElement("03");
                        model1_2.addElement("04");
                        model1_2.addElement("05");
                        model1_2.addElement("06");
                        giorno2 = new JComboBox(model1_2);

                        DefaultComboBoxModel model2_2 = new DefaultComboBoxModel();
                        model2_2.addElement("Settembre");
                        model2_2.addElement("Ottobre");
                        model2_2.addElement("Novembre");
                        model2_2.addElement("Dicembre");
                        mese2 = new JComboBox(model2_2);

                        DefaultComboBoxModel model3_2 = new DefaultComboBoxModel();
                        model3_2.addElement("2012");
                        model3_2.addElement("2013");
                        model3_2.addElement("2014");
                        anno2 = new JComboBox(model3_2);

                        dataFine = new JPanel(new FlowLayout());
                        dataFine.add(giorno2);
                        dataFine.add(mese2);
                        dataFine.add(anno2);

                        campi3.add(dataFine);

                        gruppo3.add(etichette3);
                        gruppo3.add(campi3);

                        grpEmpty=false;
                }

                contDetPrest.add(gruppo3, BorderLayout.CENTER);
        }
```

```
                                else {
                                        boolean svuota=true;

                                        flag=null;
                                int risp = JOptionPane.showConfirmDialog(fin, "Verranno rimossi i dettagli prestito",
                                "Attenzione",JOptionPane.OK_CANCEL_OPTION);
                                        if(risp==JOptionPane.CANCEL_OPTION){svuota=false; flag=true;}
                                        if(risp==JOptionPane.OK_OPTION){box.setSelected(false); flag=false;}

                                        if(svuota==true){

                                                myModOp.impostaStatoConf2(true);
                                                myModOp.conf.setEnabled(myModOp.dammiStatoConf());

                                                rb1.setSelected(true);
                                                campoMuseo.setText("");
                                                giorno.setSelectedIndex(0);
                                                mese.setSelectedIndex(0);
                                                anno.setSelectedIndex(0);
                                                giorno2.setSelectedIndex(0);
                                                mese2.setSelectedIndex(0);
                                                anno2.setSelectedIndex(0);

                                                mkl4.ripristina();

                                                contDetPrest.remove(gruppo3);
                                                contDetPrest.setBorder(BorderFactory.createEmptyBorder());
                                                fin.setSize(420,600);

                                        }

                                }

                        }

                }

}
```

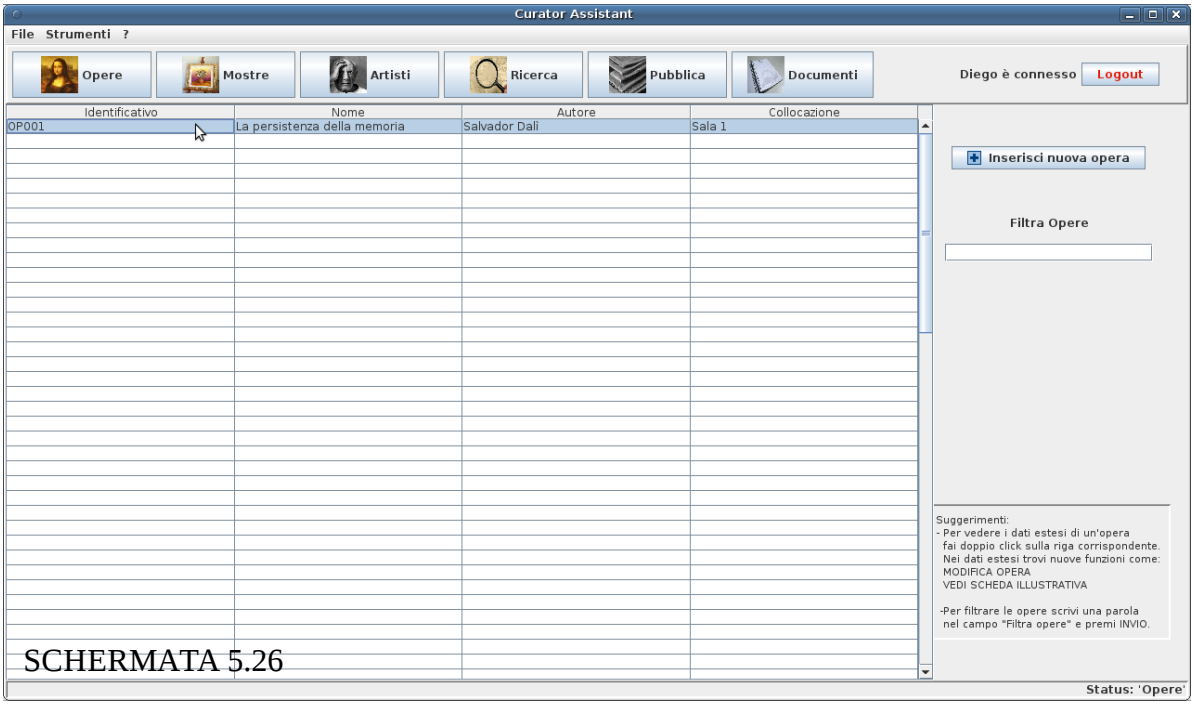
```
class MyConfActionListener3 implements ActionListener {

        DatiEstesiGUI deGUI;
        MyModOperaActionListener modOp;
        String[] ss = new String[5];

        public MyConfActionListener3(DatiEstesiGUI deG, MyModOperaActionListener mO ){deGUI=deG; modOp=mO; }

        public void actionPerformed(ActionEvent e){

                ss[0]=modOp.campoPolizza.getText();

        if(modOp.box.isSelected()==false){ if(!ss[0].isEmpty()) { deGUI.aggiungiModifiche(ss, false); }}

                else{
                        if(ss[0].isEmpty()) { ss[0] = new String("Nessuna polizza assicurativa"); }

                        if(modOp.mil2.rb1.isSelected()) {

                                ss[1] = new String("Opera in uscita");
                                ss[2] = new String("Museo cessionario");
                        }
                        else {
                                ss[1] = new String("Opera in entrata");
                                ss[2] = new String("Museo cedente");
                        }

                        ss[3]=new String(modOp.mil2.giorno.getSelectedItem().toString()
+"-"+modOp.mil2.mese.getSelectedItem().toString()+"-"+modOp.mil2.anno.getSelectedItem().toString() );
                        ss[4]=new String(modOp.mil2.giorno2.getSelectedItem().toString()
+"-"+modOp.mil2.mese2.getSelectedItem().toString()+"-"+modOp.mil2.anno2.getSelectedItem().toString() );
                        deGUI.aggiungiModifiche(ss, true);
                }
                modOp.modificaOpera.dispose();
        JOptionPane.showMessageDialog(deGUI.menuPrincipale, "Modificata opera 'La persistenza della memoria' ");
        }
}
```

```
class MyKeyListener4 implements KeyListener {

        MyModOperaActionListener modOp;
        MyItemListener2 myItem;
        MyKeyListener4 mkl4;

        Boolean sug = false;

        MyKeyListener4(MyModOperaActionListener mO, MyItemListener2 mI){modOp=mO; myItem=mI; mkl4=this;}


                        public void keyPressed(KeyEvent e){}
                        public void keyTyped(KeyEvent e){}
                        public void keyReleased(KeyEvent e){

if((myItem.campoMuseo.getText().compareTo("MoMA di New York")==0||
myItem.campoMuseo.getText().compareTo("moma")==0 )&& !modOp.campoNome.getText().isEmpty()){
                                        modOp.impostaStatoConf2(true);
                                        modOp.conf.setEnabled(modOp.dammiStatoConf());
                                        }
                                else {
                                        modOp.impostaStatoConf2(false);
                                        modOp.conf.setEnabled(modOp.dammiStatoConf());
                                        }

if(myItem.campoMuseo.getText().compareTo("m")==0||myItem.campoMuseo.getText().compareTo("mo")==0 ||
myItem.campoMuseo.getText().compareTo("mom")==0||myItem.campoMuseo.getText().compareTo("moma")==0 ) {

                                        modOp.modificaOpera.setSize(450,700);

                                        myItem.gl3.setRows(7);

                                        myItem.gruppo3.remove(myItem.etichette3);
                                        myItem.gruppo3.remove(myItem.campi3);
                                        myItem.etichette3.removeAll();

                                        myItem.etichette3.add(myItem.uscitaL);
                                        myItem.etichette3.add(myItem.entrataL);
                                        myItem.etichette3.add(myItem.cesCedL);
                                        myItem.etichette3.add(new JLabel());
                                        myItem.etichette3.add(new JLabel());
                                        myItem.etichette3.add(myItem.inizioL);
                                        myItem.etichette3.add(myItem.fineL);

                                        myItem.campi3.removeAll();

                                        DefaultListModel listmod1 = new DefaultListModel();
                                        listmod1.addElement("MoMA di New York");
                                        JList list= new JList(listmod1);

                                        list.addMouseListener(new MouseAdapter(){

                                                public void mousePressed(MouseEvent e){

                                        Object elemSel =selezionaElemento((JList)e.getSource(),e.getPoint());


                                                myItem.campoMuseo.setText(elemSel.toString());
                                                mkl4.ripristina();

if(myItem.campoMuseo.getText().compareTo("MoMa di New York")==0 && !myItem.campoMuseo.getText().isEmpty()){
                                                        modOp.impostaStatoConf2(true);
                                                        modOp.conf.setEnabled(modOp.dammiStatoConf());
                                                        }

                                                }

                                        });

                                        myItem.campi3.add(myItem.rb1);
                                        myItem.campi3.add(myItem.rb2);
                                        myItem.campi3.add(myItem.text5);
                                        myItem.campi3.add(list);
                                        myItem.campi3.add(new JLabel());
                                        myItem.campi3.add(myItem.dataInizio);
                                        myItem.campi3.add(myItem.dataFine);

                                        myItem.gruppo3.add(myItem.etichette3);
                                        myItem.gruppo3.add(myItem.campi3);
```

```
                                        myItem.text5.setBackground(new Color(255,255,255));

                                        modOp.modificaOpera.repaint();
                                        modOp.modificaOpera.setVisible(true);

                                        myItem.campoMuseo.requestFocus();

                                        sug=true;
                                }
                                else { this.ripristina(); }

                        }

                        public void ripristina() {

                                if(sug==true){
                                        myItem.gl3.setRows(5);

                                        myItem.gruppo3.remove(myItem.etichette3);
                                        myItem.gruppo3.remove(myItem.campi3);
                                        myItem.etichette3.removeAll();

                                        myItem.etichette3.add(myItem.uscitaL);
                                        myItem.etichette3.add(myItem.entrataL);
                                        myItem.etichette3.add(myItem.cesCedL);
                                        myItem.etichette3.add(myItem.inizioL);
                                        myItem.etichette3.add(myItem.fineL);

                                        myItem.campi3.removeAll();


                                        myItem.campi3.add(myItem.rb1);
                                        myItem.campi3.add(myItem.rb2);
                                        myItem.campi3.add(myItem.text5);
                                        myItem.campi3.add(myItem.dataInizio);
                                        myItem.campi3.add(myItem.dataFine);

                                        myItem.gruppo3.add(myItem.etichette3);
                                        myItem.gruppo3.add(myItem.campi3);

                                        myItem.text5.setBackground(myItem.text6.getBackground());
if((myItem.campoMuseo.getText().compareTo("MoMA di New York")==0||
myItem.campoMuseo.getText().compareTo("moma")==0 )&& !modOp.campoNome.getText().isEmpty()){
                                                modOp.impostaStatoConf2(true);
                                                modOp.conf.setEnabled(modOp.dammiStatoConf());
                                                }

                                        modOp.modificaOpera.setSize(450,650);
                                        modOp.modificaOpera.repaint();
                                        modOp.modificaOpera.setVisible(true);

                                        myItem.campoMuseo.requestFocus();

                                        sug=false;
                                }

                        }

                        public static Object selezionaElemento(JList l, Point clickPoint){

                                int index= l.locationToIndex(clickPoint);
                                if (index >=0) {  //se non lo e', no elemento lista
                                        l.setSelectedIndex(index);
                                        Object selezionato= l.getModel().getElementAt(index);
                                        return selezionato;
                                }
                                else return null;

                        }

}
```

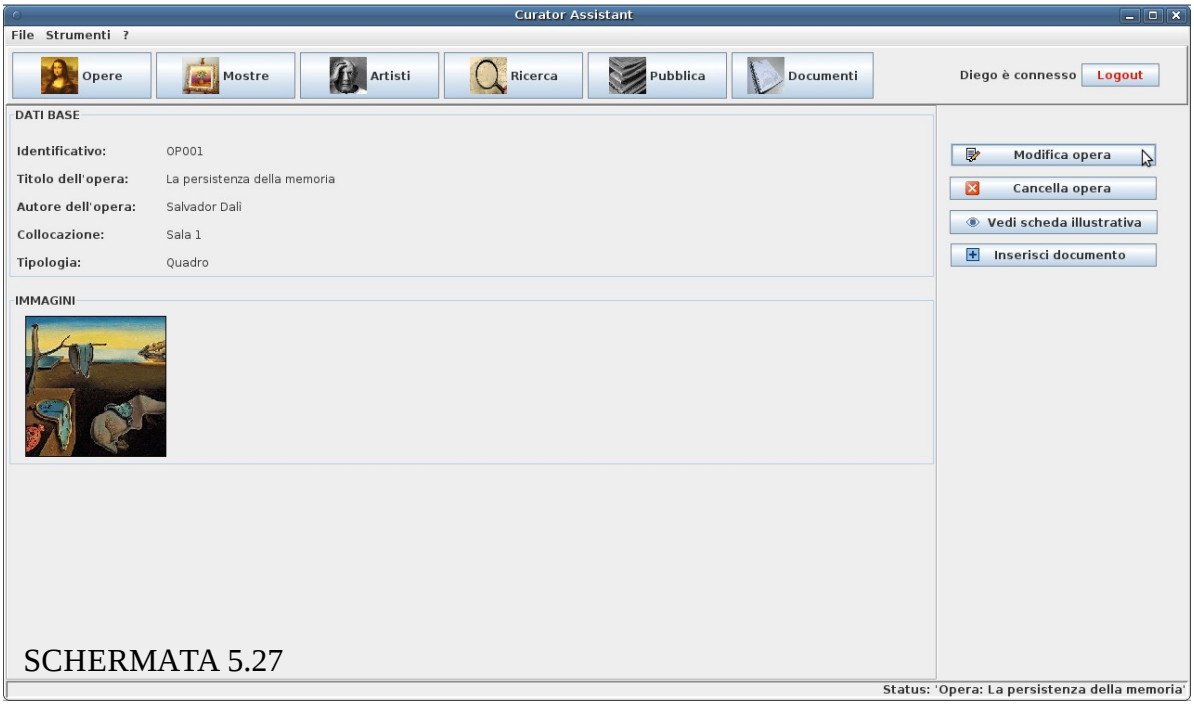
5.5.4 GUI “Registrare prestito”

Diego fa doppio click sull'opera interessata, nella pagina dedicata alle 'Opere':



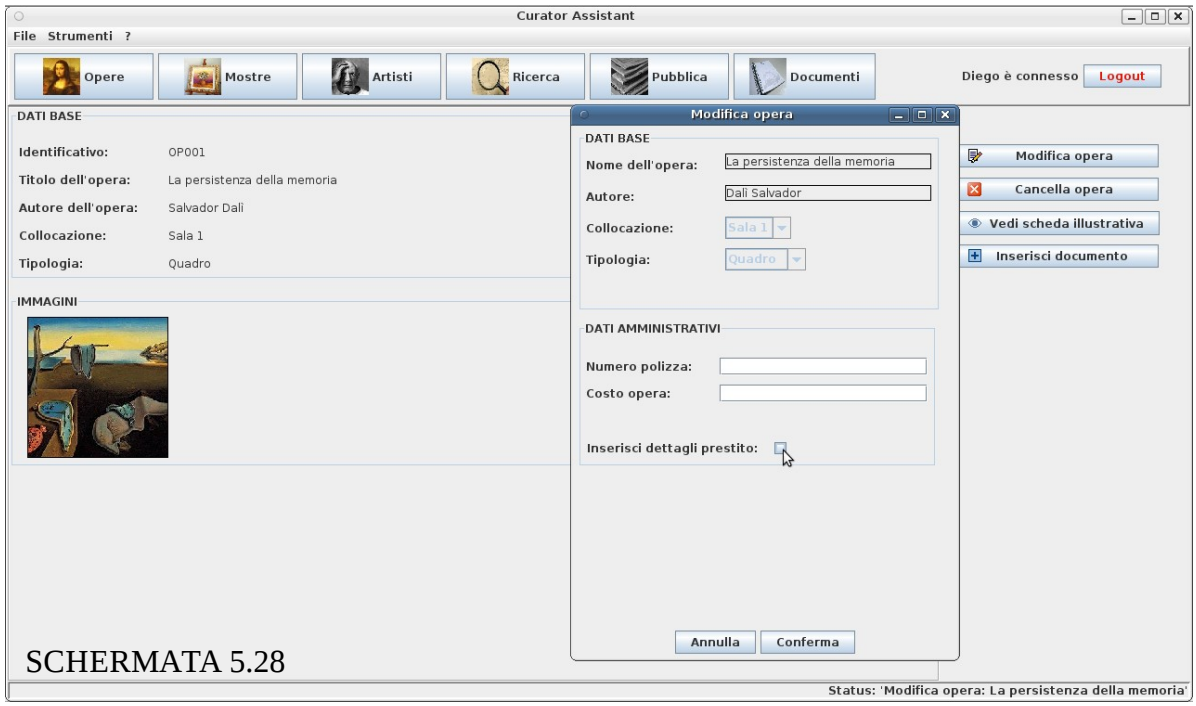
SCHERMATA 5.26

Ottiene i dati dell'opera per esteso, quindi clicca su 'Modifica opera':



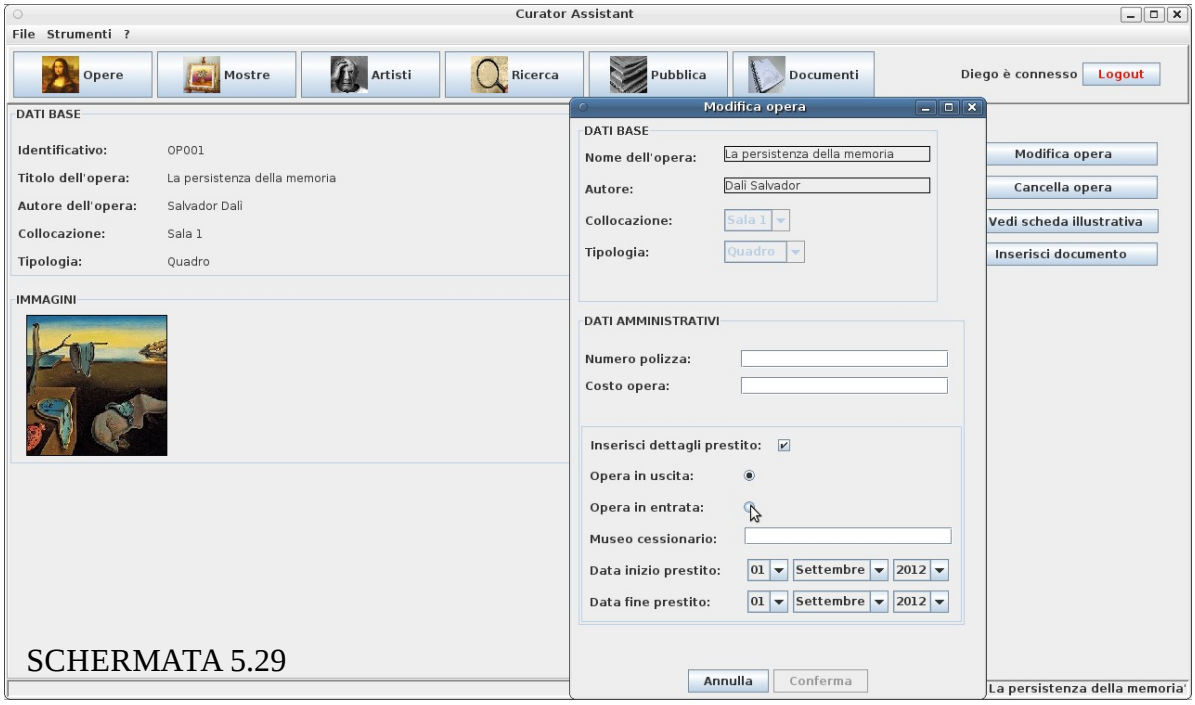
SCHERMATA 5.27

Nella finestra dedicata alla modifica dell'opera Diego seleziona la voce 'Inserisci dettagli prestito':



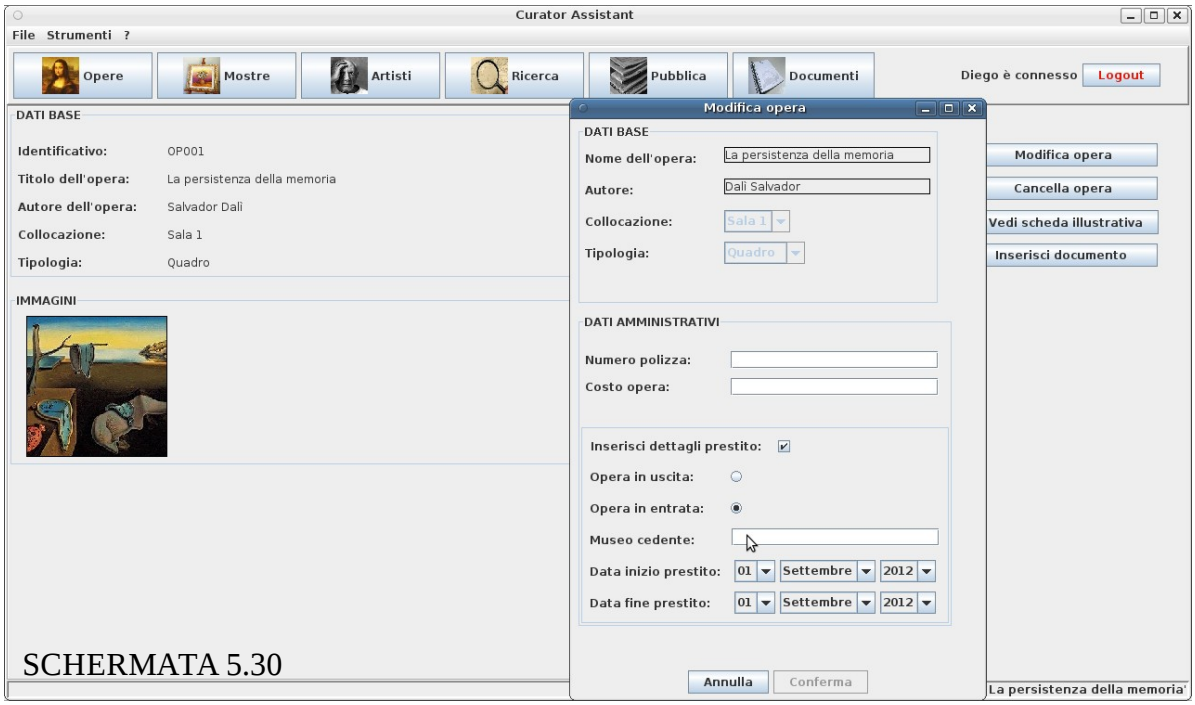
SCHERMATA 5.28

La finestra si espande permettendo l'inserimento delle informazioni sull'eventuale prestito in cui è coinvolta l'opera, in particolare Diego specifica che l'opera è in 'Entrata' cioè è stata prestata al museo.



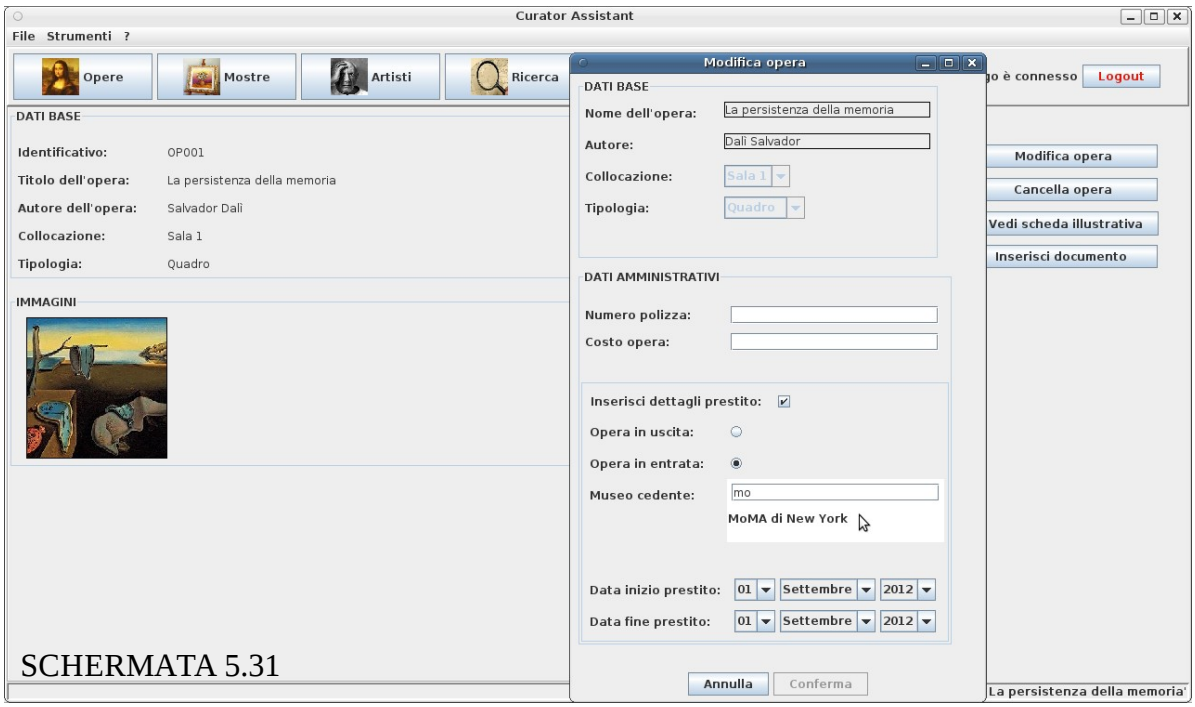
SCHERMATA 5.29

Selezionando 'Opera in entrata' la finestra si aggiorna permettendo l'inserimento del 'Museo cedente' (inizialmente veniva richiesto l'inserimento del 'Museo cessionario' perché di *default* era selezionato 'Opera in uscita').



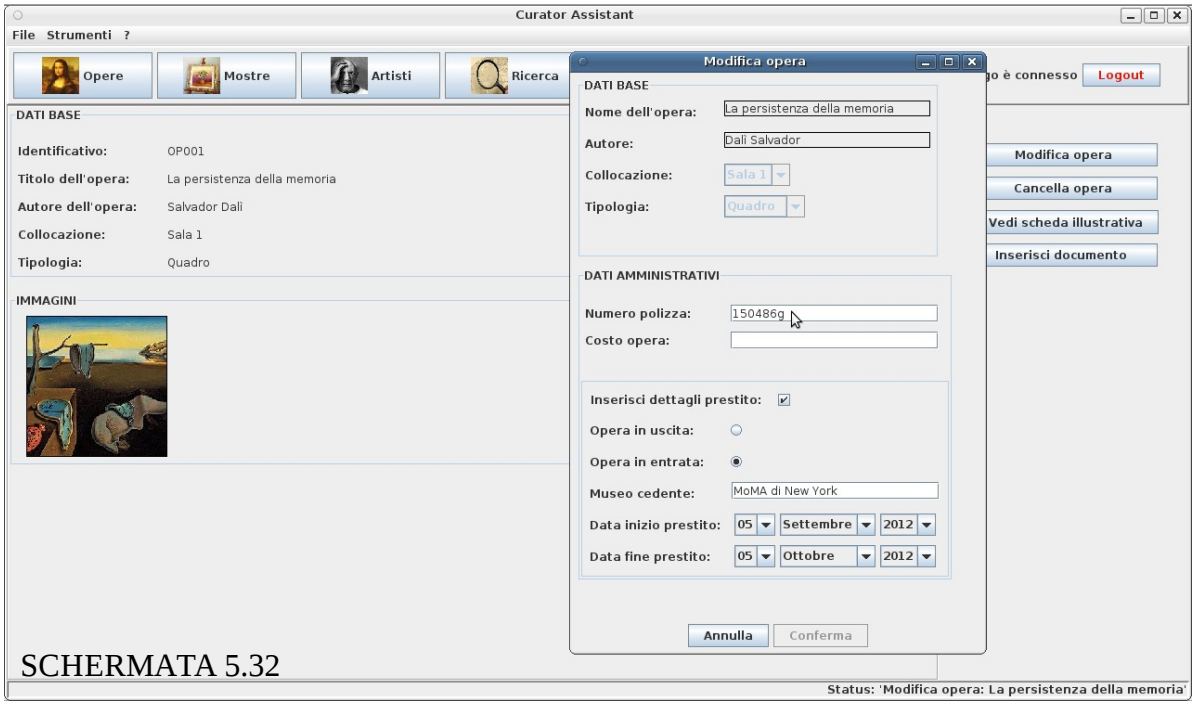
SCHERMATA 5.30

Mentre Diego scrive il 'Museo cedente' il sistema gli suggerisce l'elenco di musei presenti nel sistema che corrisponde a quanto digitato, Diego riconosce il museo che vuole inserire e lo seleziona:



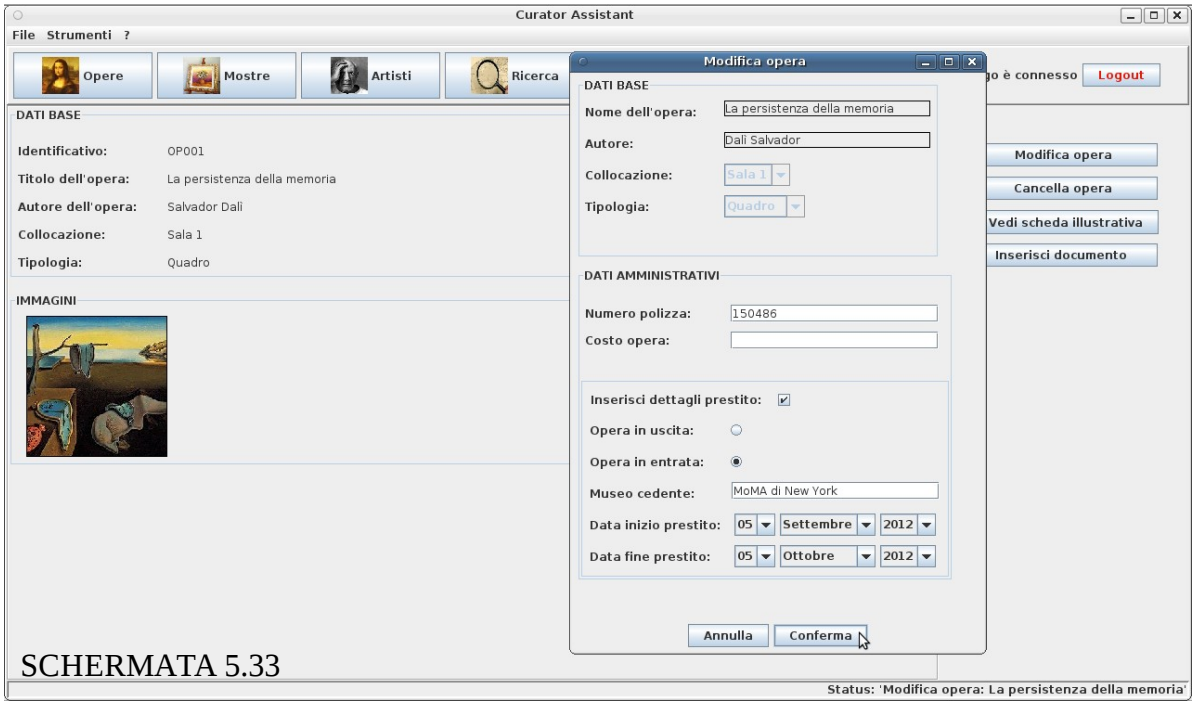
SCHERMATA 5.31

Dopo aver finito di inserire i dettagli sul prestito, Diego scrive il numero della polizza con cui è stato assicurato il prestito dell'opera. Per sbaglio Diego digita una lettera e il sistema non permette la conferma dei dati visto che il 'Numero polizza' deve essere un valore intero.



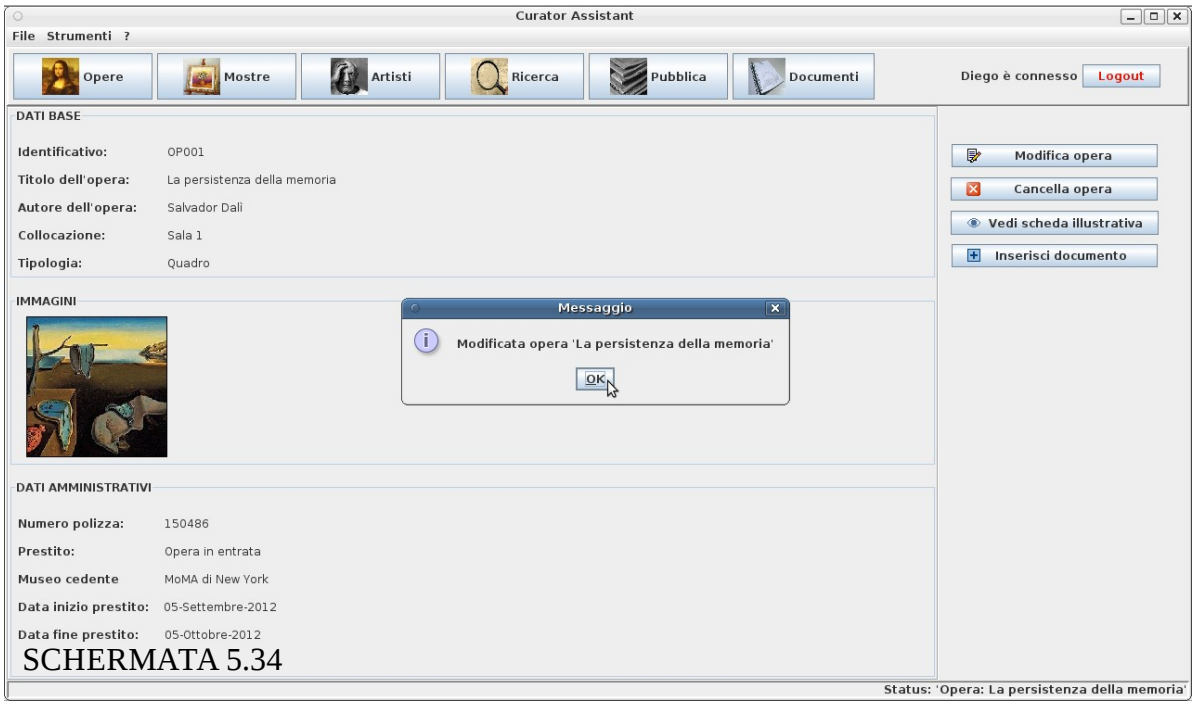
SCHERMATA 5.32

Accorgendosi dell'errore Diego corregge il numero della polizza e conferma i dati:



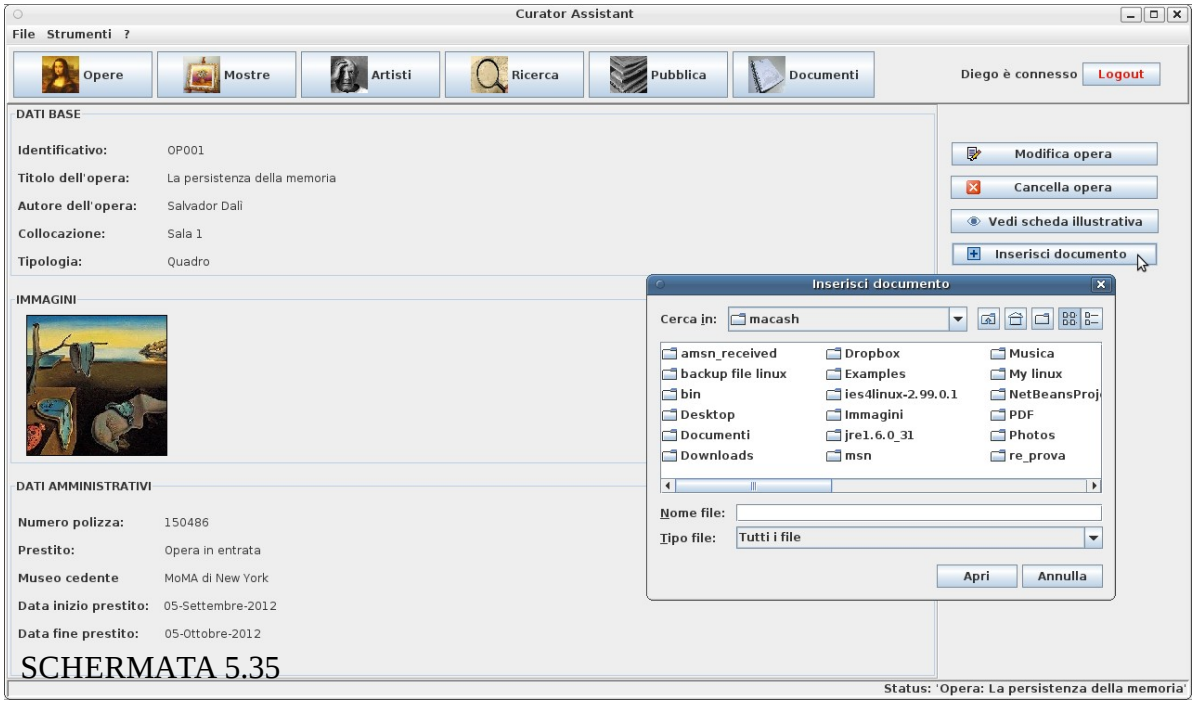
SCHERMATA 5.33

I dati amministrativi vengono aggiunti all'opera e visualizzati nella relativa pagina:



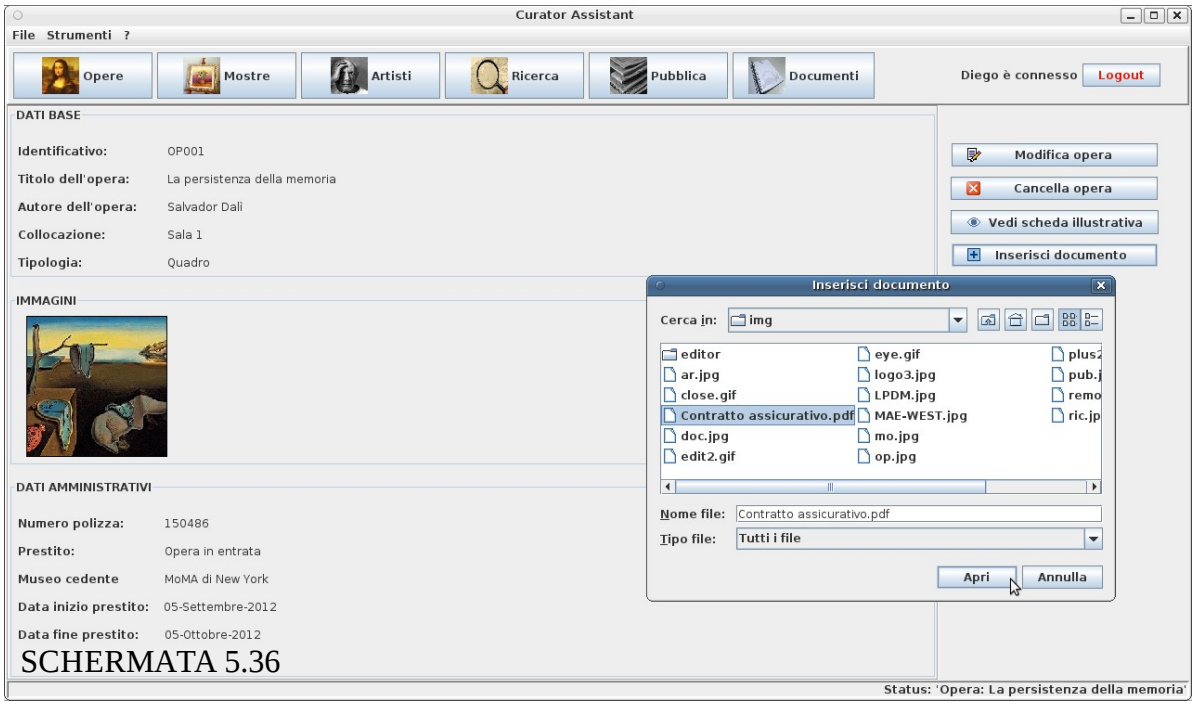
SCHERMATA 5.34

A questo punto Diego clicca su 'Inserisci documento' per inserire il contratto dell'assicurazione dell'opera in formato PDF:



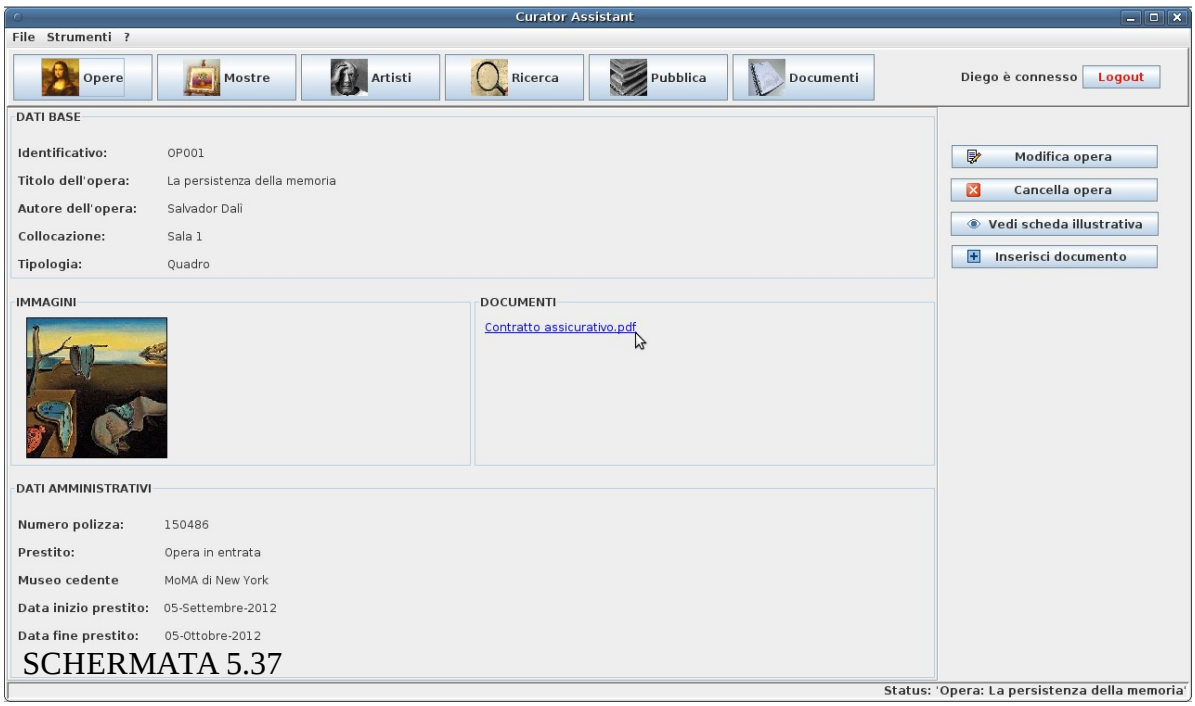
SCHERMATA 5.35

Diego seleziona il file tra le cartelle del suo *personal computer*:



SCHERMATA 5.36

Il documento rimane associato all'opera come è possibile vedere dalla relativa pagina dei 'Dati estesi':



SCHERMATA 5.37

5.5.5 Valutazione “Registrare prestito”

Visibilità dello stato del sistema:

Oltre agli accorgimenti utilizzati nelle sezioni di interfaccia precedenti (barra di stato ecc.) in questa sezione possiamo notare ulteriori accorgimenti ad esempio, nella schermata 5.29 e 5.30 quando l'utente seleziona 'Opera in entrata' il sistema modifica il suo stato e lo rende visibile all'utente modificando l'etichetta del campo successivo da 'Museo cessionario' a 'Museo cedente'. Inoltre dopo l'inserimento dei dati amministrativi (schermata 5.34) il sistema fornisce dei feedback del cambiamento del suo stato mediante una *dialog* che conferma all'utente che l'opera è stata modificata e aggiornando la pagina dei 'Dati estesi' dell'opera che adesso contiene oltre ai 'Dati base' anche i 'Dati amministrativi' appena inseriti. Stessa cosa avviene quando l'utente inserisce la copia PDF del contratto assicurativo (schermata 5.35 – 5.37) il sistema fornisce un feedback dell'inserimento del documento aggiornando la pagina dei 'Dati estesi' dell'opera dove adesso risulta presente anche l'area dedicata ai documenti (schermata 5.37).

Corrispondenza fra il mondo reale e il sistema:

Il sistema cerca sempre di “parlare” il linguaggio dell'utente, in questo caso l'utente tipo è Diego che come sappiamo non ha competenze artistico letterarie pertanto possedendo un account di tipo amministrativo quando accede alla sezione 'Modifica opera' non ha potere di controllo sui 'Dati base' dell'opera che possono richiedere competenze museali (schermata 5.28) mentre può operare sui campi amministrativi dove ci sono concetti vicini al suo profilo (campo 'Numero polizza' ecc.)

Libertà e controllo da parte degli utenti:

Come sempre l'utente può inserire i dati nell'ordine che vuole, nelle schermate 5.31 – 5.33 si vede chiaramente che l'utente inserisce prima i dettagli sul prestito e poi il numero della polizza assicurativa anche se le informazioni sono presentate nell'ordine inverso. Tutte le operazioni sono reversibili, se l'utente non ha più intenzione di inserire i dettagli sul prestito basta che deselezioni la relativa *checkbox*, se vuole annullare l'intera modifica dell'opera o l'inserimento di un documento

basta che utilizzi il pulsante 'Annulla' nelle corrispettive finestre.

Consistenza e standard:

Anche se adesso l'utente ha un account da amministratore, la struttura delle finestre 'Modifica opera' e 'Inserisci documento' rimane coerente a quanto visto nei paragrafi precedenti adottando una struttura tipica delle più comuni applicazioni a finestre. Si utilizzano componenti standard come *checkbox*, *combobox*, *radio button*, che l'utente ha imparato ad usare in altre applicazioni e la finestra 'Inserisci documento' della schermata 5.35 (come del resto la finestra 'Inserisci immagine' schermata 5.21) segue le convenzioni di un classico selettore di file.

Prevenzione degli errori:

Come nelle interfacce precedenti, l'utente non può confermare la modifica dell'opera se non ha inserito correttamente i nuovi dati. In particolare nella schermata 5.28 si può vedere che se l'utente non modifica alcun dato allora può confermare perché i dati precedenti erano già corretti; se seleziona 'Inserisci dettagli prestito' non può confermare finché non inserisce il museo cedente (o cessionario a seconda dei casi) perché questa informazione è considerata obbligatoria per i dettagli sul prestito (Schermata 5.29). Di nuovo l'utente non può confermare se non inserisce un valore numerico nel campo 'Numero polizza' (Schermata 5.32). Le *checkbox* costringono l'utente a inserire solo date corrette e i suggerimenti per il campo 'Museo cedente/cessionario' lo aiutano a inserire solo musei validi. Come sempre tutto ciò previene l'utente a procedere in situazioni lo che portino in errore.

Riconoscere piuttosto che ricordare:

Nella finestra 'Modifica opera' sono accessibili funzioni che l'utente potrebbe non usare; nel caso specifico 'Inserisci dettagli prestito' non viene selezionato se l'opera non è coinvolta in un prestito e, ad esempio, l'utente vuole inserire come dato amministrativo il numero della polizza assicurativa sul furto di un'opera della collezione permanente. Tuttavia si è scelto di rendere visibile l'opzione 'Inserisci dettagli prestito' nella finestra appunto perché l'utente possa riconoscerla quando ne ha

bisogno invece che ricordare dove trovarla se nascosta in un menu più complesso. Stessa cosa nella schermata 5.27 sono resi ben visibili le operazioni più comuni su di un'opera attraverso i pulsanti (Modifica opera, Cancella opera ecc.)

Flessibilità ed efficienza d'uso:

La finestra 'Modifica opera' così come la visualizza un utente *amministratore* ha un aspetto più complesso rispetto a come visualizza la finestra 'Inserisci/Modifica opera' un utente *curatore*, complessità che aumenta ulteriormente quando si selezionano i dettagli sul prestito. Questo perché si presuppone che l'utente amministratore sia più esperto nell'uso di un sistema informatico; quindi il sistema cerca di adattarsi al tipo di utente che si è autenticato.

Design minimalista:

Come abbiamo detto precedentemente l'utente potrebbe non usare l'opzione 'Inserisci dettagli prestito' nella finestra 'Modifica opera', tale opzione però non è stata eliminata dalla finestra perché potrebbe essere utilizzata non di rado. Tuttavia quando questa opzione è deselezionata i campi relativi al prestito non sono visibili in modo che il dialogo sia minimale soltanto al tipo di inserimento che l'utente vuole effettuare.

Aiutare gli utenti a riconoscere gli errori, diagnosticarli e correggerli:

L'utente tipo di questa parte di interfaccia è per noi Diego, dal suo profilo vediamo che non riesce a rimanere concentrato per periodi lunghi di tempo pertanto può commettere errori di distrazione come ad esempio l'errore di digitazione della schermata 5.32. In questo caso il sistema non permette la conferma dei dati ma se l'utente non si accorge del motivo per cui non può confermare i dati deve essere previsto un sistema di notifica che visualizzi in un messaggio di fianco al campo perché l'inserimento per quel campo è invalido come accade nella schermata 5.16. Ciò aiuta l'utente a riconoscere la situazione di errore (nel codice di paragrafo 5.5.3 tali notifiche non sono state implementate per non aumentare la complessità di implementazione, in ogni caso tale implementazione è del tutto simile a quella usata nel paragrafo 5.4.3).