CURATOR ASSISTANT®

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

Identificativo del progetto: CA
Nome documento: System Design(SD)
Identificativo del documento: 6-CA_SD_E1_R1
Data del documento: 21/05/2012
Prima revisione del documento: 04/06/2012

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

1 Introduzione

1.1 Obiettivi

L'obbiettivo del documento è quello di descrivere l'attività di *design* svolta per il progetto. Nel *design* lo scopo è quello di porre soluzioni concrete al modello di analisi (e cioè aggiungere il 'come' al 'cosa'); verrà quindi aggiunto ciò che serve affinché i requisiti formalizzati nel documento di analisi **5_CA_SA** possano essere effettivamente implementati e nel particolare:

- Sanciremo l'architettura definitiva su cui si poggerà il sistema.
- Evolveremo le *classi di analisi* in *classi di design* raffinando le associazioni tra classi, aggiungendo classi di supporto (per GUI, database, ecc.) e definendo concretamente attributi e metodi delle classi.
- Rivedremo l'interazione tra classi per realizzare alcuni casi d'uso del documento **4_CA_UCM**, questa volta utilizzando le classi di design (compreso classi di supporto) il che farà vedere come il caso d'uso viene effettivamente risolto.

1.2 Identificativo documento

Questo documento è identificato con il codice **6-CA_SD_E1_R1** 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** .

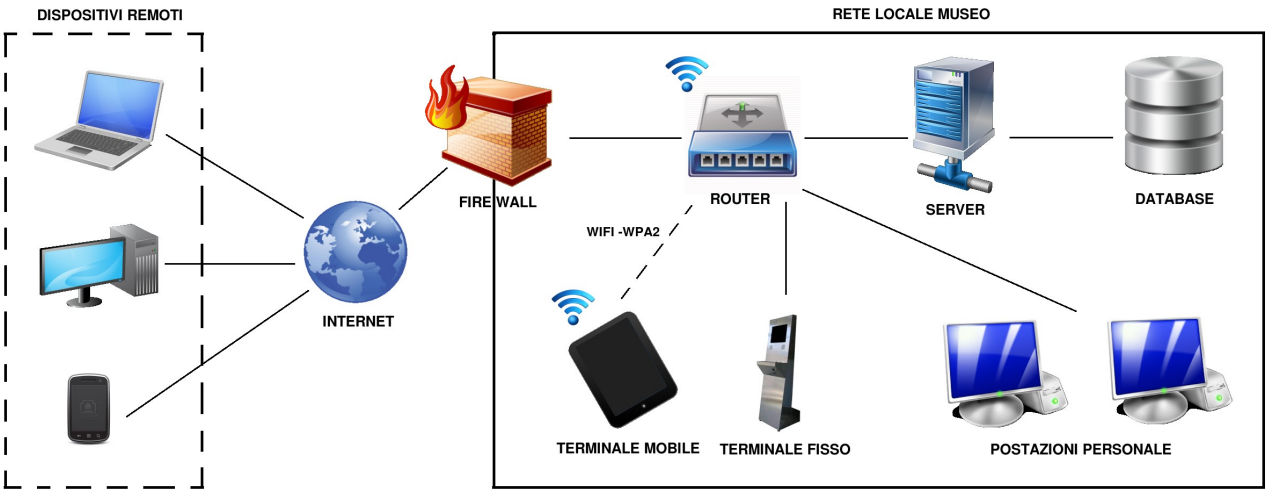
2 Progetto Architetturale

2.1 Comunicazione fisica

L'architettura fisica si compone di una rete locale IEEE 802.3 già presente nel distretto del museo su cui verrà installato un nuovo server fisico e un nuovo dispositivo di *storage* per la base di dati. Alla rete sono già connessi i *personal computer* del personale del museo mentre nuovi terminali fissi saranno collegati nelle varie sale del museo. Terminali mobili potranno comunicare con la rete tramite lo standard IEEE 802.11 e l'accesso alla rete Wi-Fi sarà consentito solo ad apparecchiature del museo con meccanismo di protezione WPA2. La rete locale comunica con l'esterno attraverso la rete internet con la quale dispositivi remoti potranno inviare richieste al server del museo; le richieste saranno monitorate e filtrate da un apparato *firewall*.

Richieste esterne (dal web) e interne (terminali, postazioni personale) sono veicolate al server da un router; il server interrogherà il database fisico per rispondere alle varie richieste e il router instraderà le risposte del server all'effettivo destinatario.

Il tutto è meglio esplicitato nello schema sequente:



2.2 Comunicazione software

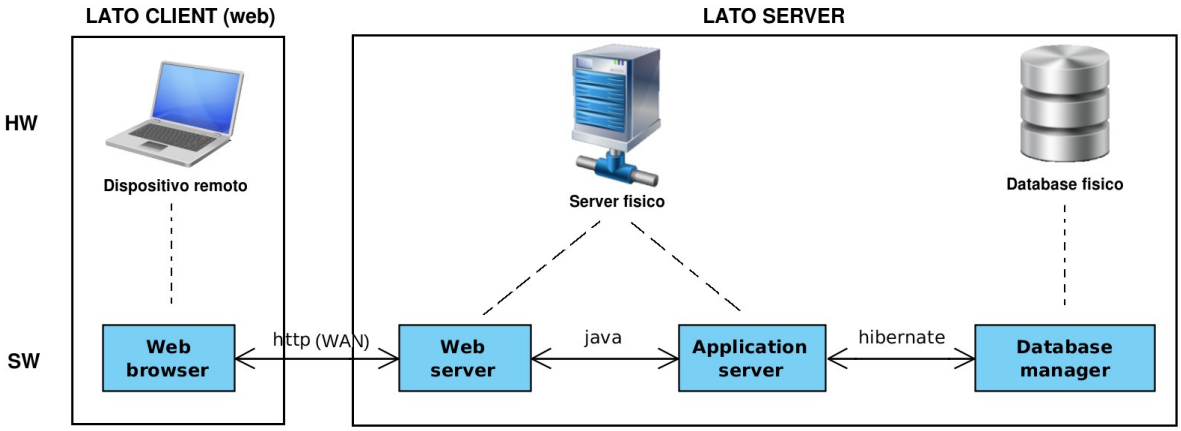
L'architettura software scelta per il progetto è di tipo client/server a più livelli; il lato server infatti si compone di più strati software separati con compiti specifici: avremo un *application server* dove si troverà la logica dell'applicazione, un *web serve*r che ascolterà le richieste provenienti dalla rete e un *database manager* che gestirà l'accesso al database. I strati software del lato server possono trovarsi sullo stesso dispositivo fisico o su dispositivi diversi (negli schemi a pag. 6 viene mostrata anche la collocazione fisica dei vari moduli software) in particolare:

- Il *web server* e *l'application server* risiedono sul server fisico e sono un'applicazione JAVA
- Il *database manager* risiede sul database fisico ed è un DBMS MySQL
- L'*application server* comunica con il *database manager* attraverso la piattaforma Hibernate che fornisce un servizio di *Object-relational mapping* (ORM) con cui effettueremo un *mapping* tra classi Java e le tabelle della base di dati. L'applicazione java userà le API di Hibernate per gestire la persistenza dei dati e in linea di principio sarà esonerata dall'esecuzione diretta di *query* SQL. L'*application serve*r avrà opportune classi DAO (Data Access Object) che forniscono un'astrazione della base di dati alle altre classi java. Le classi DAO nella maggior parte dei casi chiameranno le API di Hibernate ma non è esclusa la richiesta di una specifica *query* SQL nei casi in cui il *mapping* non risulta sufficiente (ad es. ricerche non di tipo 'diretto') nell'approccio DAO classico. Inoltre Hibernate fornisce indipendenza dal tipo di DBMS usato che potrà essere variato senza apportare modifiche all'applicazione java.

Per quanto riguarda il lato client si adotteranno tecnologie differenti a seconda del tipo di client (web, terminali, client dedicato) esplicitate di seguito:

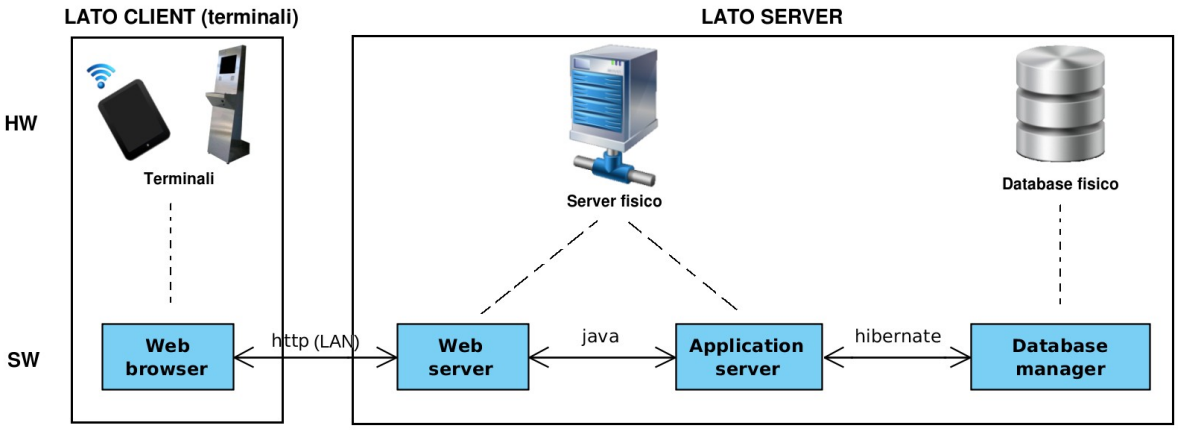
2.2.1 Web

Per quanto riguarda il *client web* avremo un'interfaccia fornita da *browser* che risiederà chiaramente sul dispositivo remoto in questione. Tramite il protocollo *http* si comunica al *web server* che servirà le varie richieste utilizzando la logica dell'*application server*.



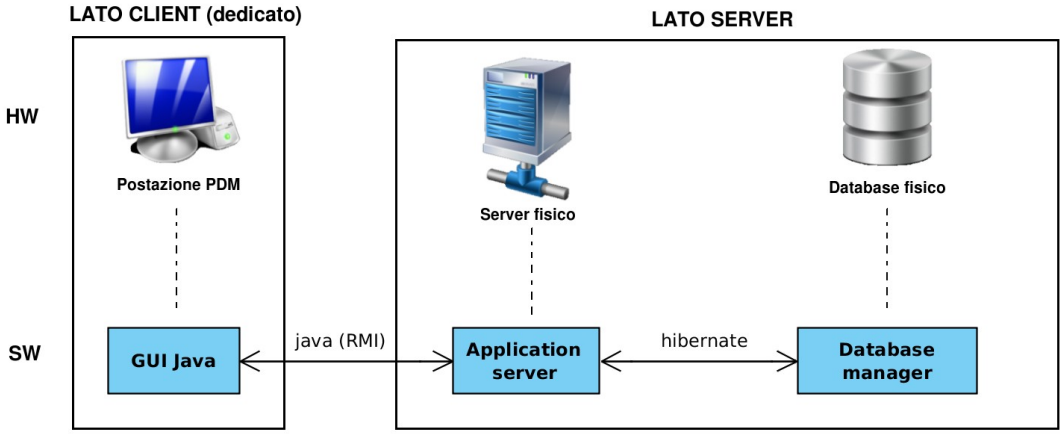
2.2.2 Terminali

La comunicazione tramite i terminali del museo adotta la stessa tecnologia della comunicazione web. I terminali saranno dotati di un'interfaccia *browser* che comunica con il *web server* in *http* similmente a quanto visto prima. La differenza si vede al livello fisico, le richieste dei terminali arrivano al *router* del museo senza passare per la rete internet e la comunicazione rimane vincolata alla sola rete locale del museo. Ovviamente l'interfaccia web dei terminali mobili sarà adattata alle caratteristiche del dispositivo.



2.2.3 Client dedicato

La comunicazione attraverso le postazioni del personale del museo adotta una tecnologia differente da quelle viste prima. Questo perché il personale è fornito di un *client dedicato* è cioè di un'applicazione java che verrà installata sulle macchine utilizzate dal personale. Tale applicazione è essenzialmente un'interfaccia grafica java che può richiamare metodi di oggetti che non sono presenti fisicamente sulle macchine delle postazioni e questo viene attuato tramite la tecnologia RMI (Remote Method Invocation). Tale tecnologia include infatti della API che rendono trasparente la comunicazione su rete e in questo modo la GUI java invoca il metodo di un oggetto remoto (nel nostro caso un oggetto dell'*application server* sul server fisico) come se l'oggetto appartenesse al *client dedicato*. Così facendo le richieste provenienti dalle postazioni PDM vengono servite come se la logica dell'applicazione risiedesse sulle macchine delle postazioni stesse.

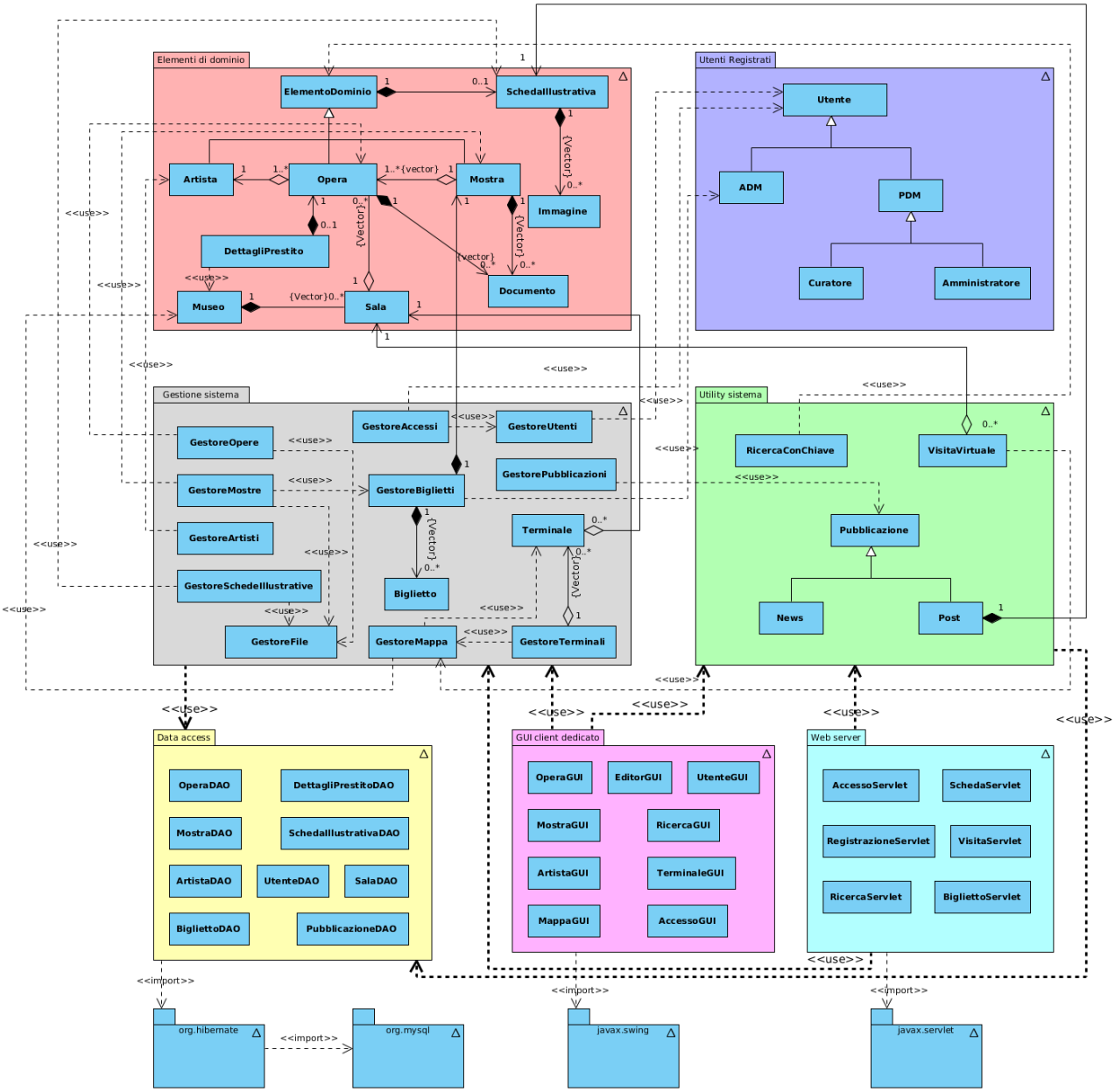


3 Classi di design

3.1 Modello delle classi (design)

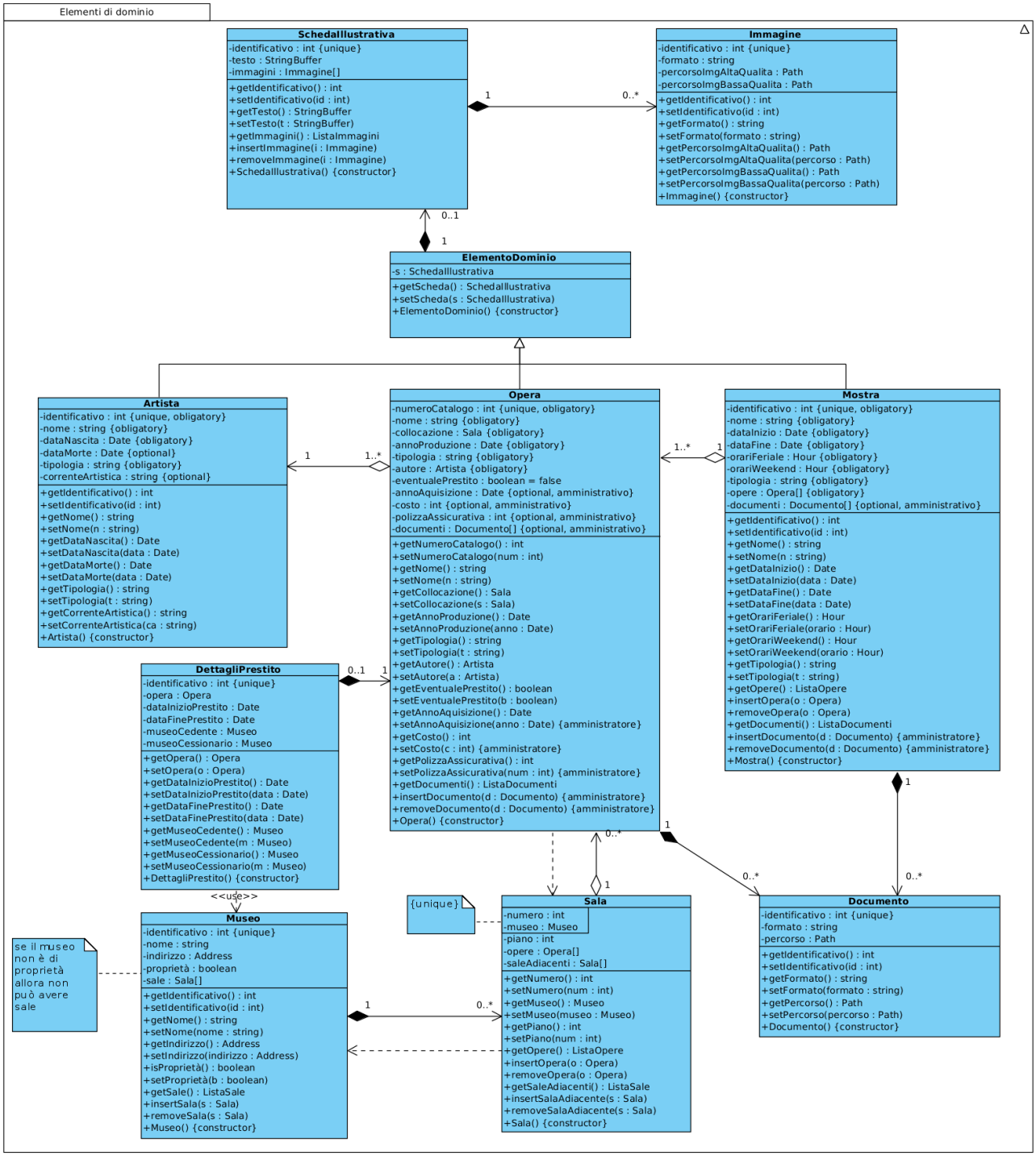
Rispetto al modello di analisi aggiungeremo le classi di supporto per la realizzazione concreta dell'architettura stabilita. Quindi avremo le classi DAO che forniranno l'interfaccia al database per gestire la persistenza dei dati, le classi Servlet per la generazione di pagine web a seconda delle richieste spedite dal browser, le classi GUI per implementare l'interfaccia grafica del *client dedicato*.

Inoltre, per un maggior supporto del modello al linguaggio di programmazione java, opereremo un raffinamento delle associazioni tramutandole in aggregazioni o composizioni.

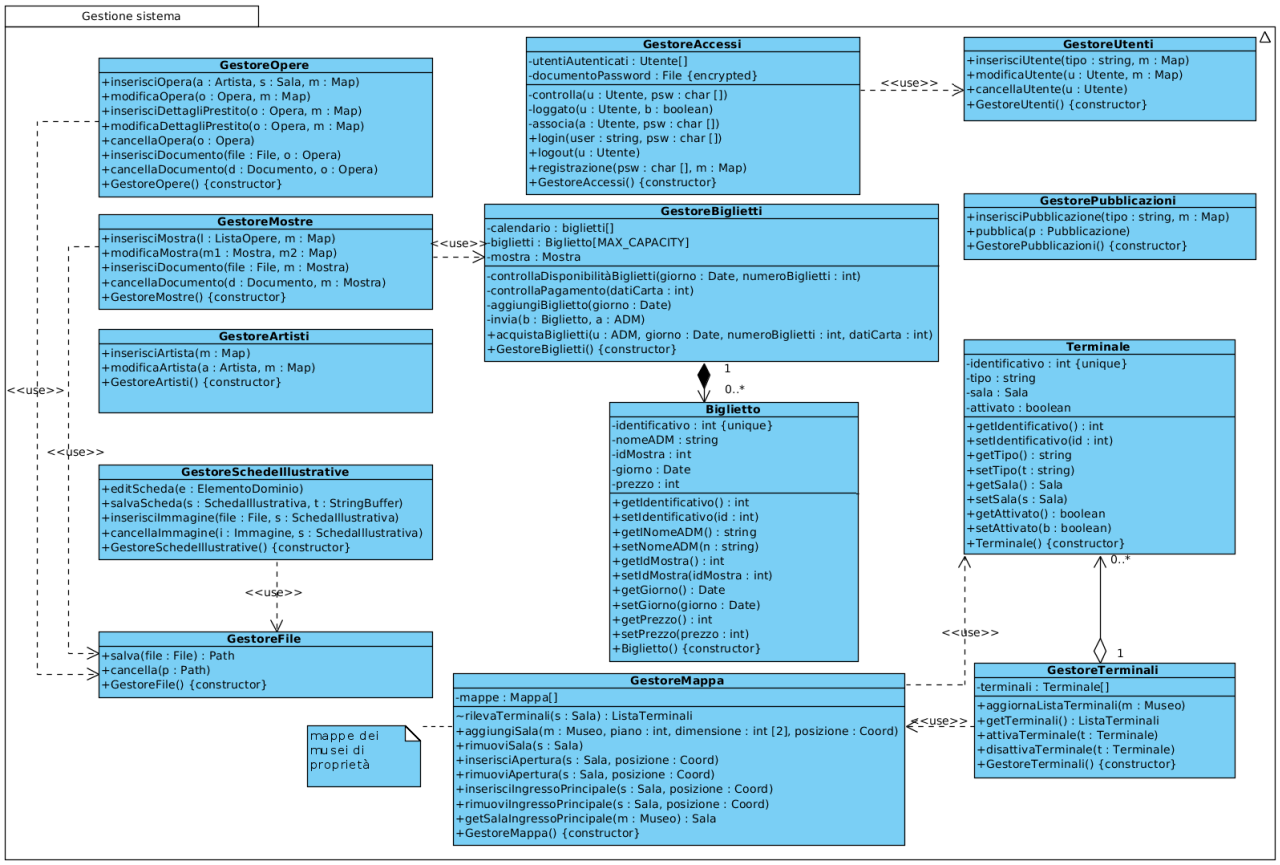


3.2 Package Elementi di dominio

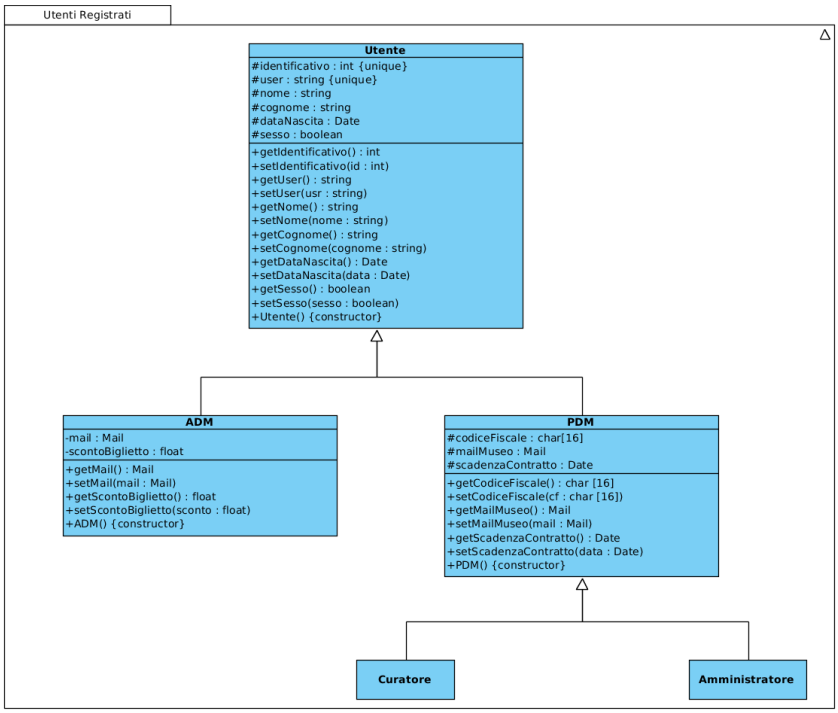
Nella descrizione dettagliata dei package rispetto all'analisi aggiungeremo il 'tipo' effettivo ad attributi e parametri dei metodi nonché il 'tipo' di oggetto ritornato da un determinato metodo, quando il 'tipo' di oggetto ritornato non è presente si presuppone di tornare *void*. Alcuni tipi di dato generici nell'analisi (Dati[], Testo, ecc..) saranno sostituiti con tipi concreti (Map, StringBuffer, ecc.) disponibili nel package java.util. Inoltre saranno aggiunti i costruttori.



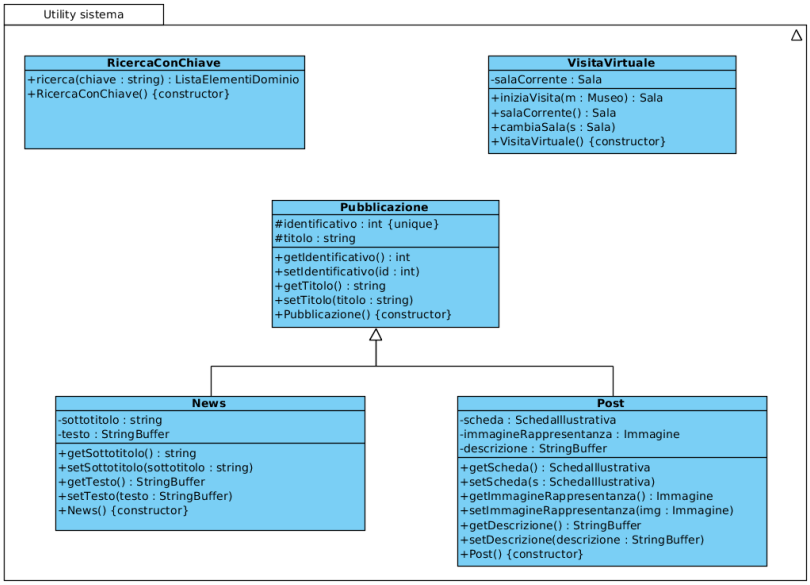
3.3 Package Gestione sistema



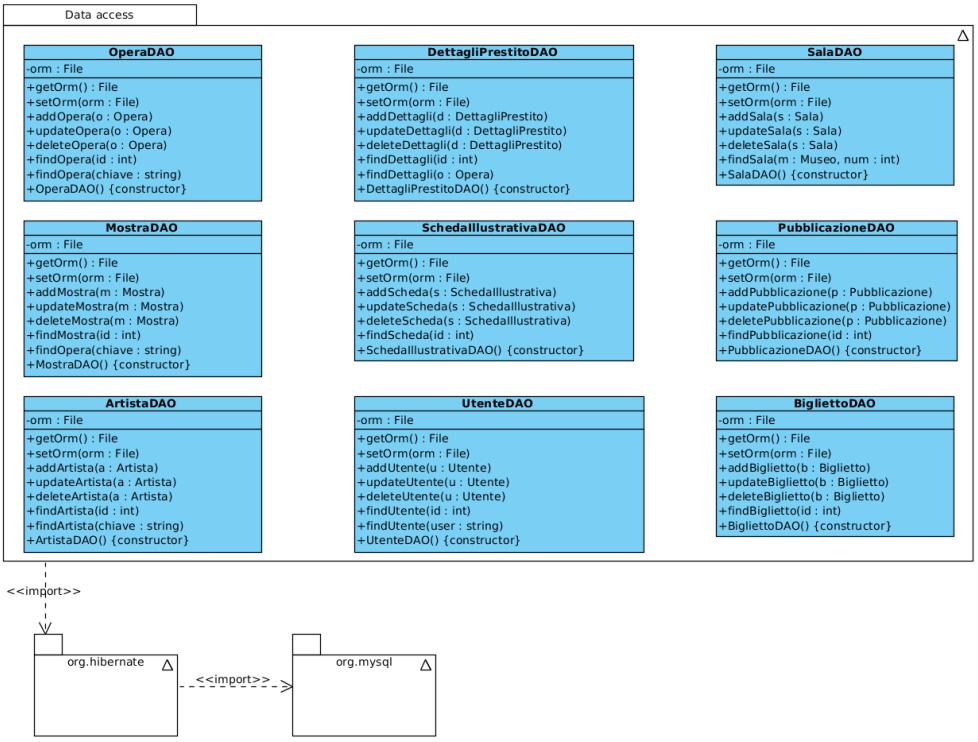
3.4 Package Utenti



3.5 Package Utility



3.6 Package Data access



4 Realizzazione dei casi d'uso (Design)

Come nell'attività di analisi, la realizzazione dei casi d'uso mostra l'interazione tra gli oggetti istanziati dalle varie classi al fine di realizzare alcuni casi d'uso del documento **4-CA_UCM** (quelli determinanti nell'architettura stabilita). Tuttavia in questo contesto, le astrazioni fatte al livello di analisi, come i *multiobject,* devono essere risolte; nel caso dei multiobject che rappresentavano insiemi di oggetti, essi verranno sostituiti con i Data Access Object (DAO) i quali si interfacciano al 'vero' insieme di dati e cioè quello del database nella gestione della persistenza. Inoltre nell'analisi si era supposto che richieste simili da parte di 'Utenti PDM' e 'Utenti pubblici' producano lo stesso scambio di messaggi tra oggetti: questo è vero quando si considera solo il lato server ma, come visto nel progetto architetturale (capitolo 2), considerando il lato *client* la tecnologia della comunicazione è differente. Quindi dal momento che nel modello delle classi di design sono state aggiunte le classi di supporto per l'implementazione dell'architettura stabilita, adesso nella realizzazione di un caso d'uso interagiranno anche gli oggetti di tali classi (classi GUI, classi Data Access, classi Servlet) e si vedrà la realizzazione concreta del caso d'uso preso in esame. Per questi motivi i diagrammi nell'analisi in cui venivano unificati 'attori' diversi verranno disgiunti e in particolare avremmo diagrammi che interessano il *client dedicato* e diagrammi che interessano il *web*. Inoltre costruiremo dei diagrammi che faranno vedere degli esempi di comunicazione tra un DAO ed gli oggetti definiti da Hibernate.

Useremo di nuovo *diagrammi di sequenza* e *diagrammi di comunicazione* ma con l'aggiunta dei livelli GUI, Data Access, Servlet. La comunicazione con la GUI Java (nel caso del *client dedicato*) e la comunicazione con le Servlet (nel caso del *web*) verrà descritta tramite stereotipi definiti di seguito. Ovviamente l'uso dei metodi delle classi all'interno dei diagrammi sarà coerente con le classi di design e saranno esplicitati i costruttori ove necessario.

L'identificativo dei diagrammi avrà la stessa struttura vista nel documento di analisi **5-CA_SA**, con l'unica differenza che conterranno la sigla 'D' (che sta per design) anziché 'A', appunto per distinguerli con i diagrammi di analisi. Non saranno convertiti tutti i diagrammi di analisi in diagrammi di design ma la traduzione dei diagrammi mancanti segue linearmente le regole dei diagrammi seguenti:

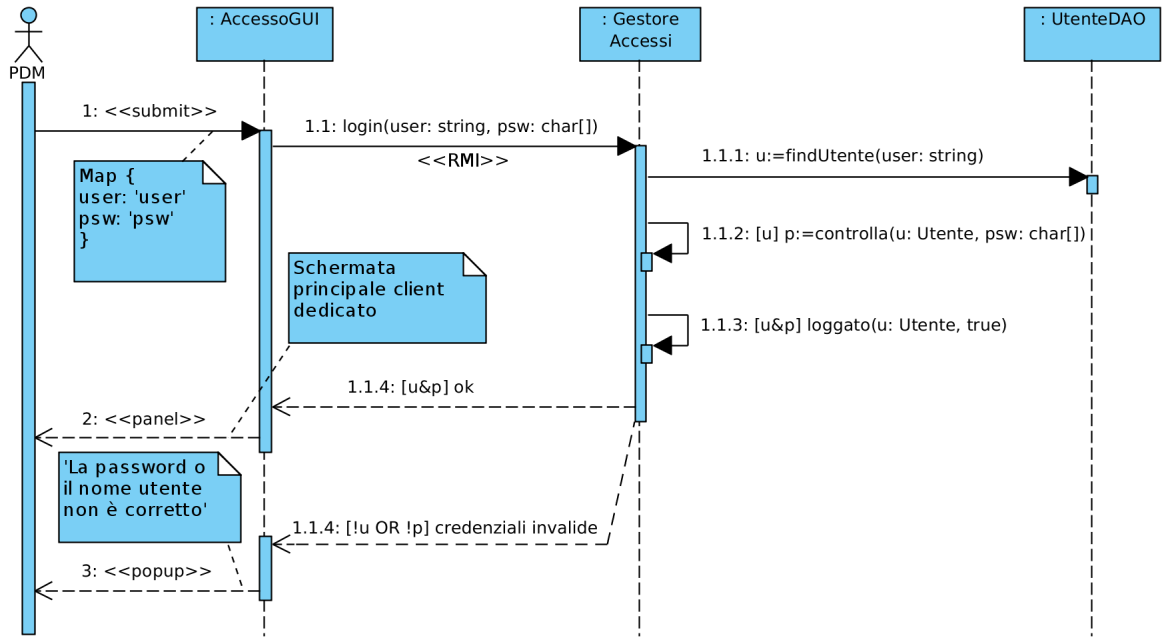
4.1 Diagrammi di sequenza / comunicazione (Design)

4.1.1 Client Dedicato

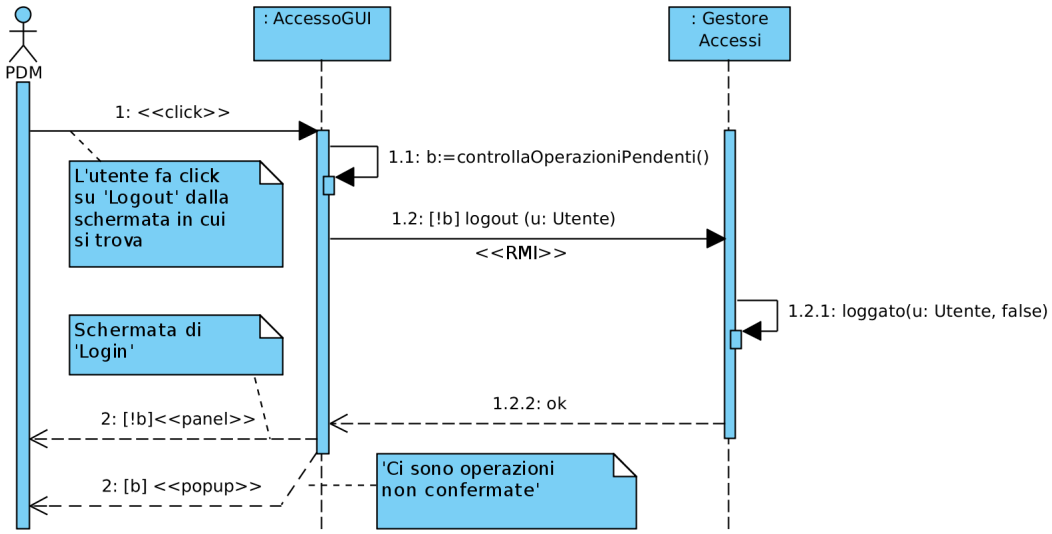
Stereotipi utilizzati:

- <<submit>> L'utente PDM sottomette la compilazione di un *form* alla GUI Java del proprio *client* dedicato. La GUI utilizza a tal proposito una oggetto di tipo Map (disponibile in java.util) che mantiene l'associazione tra il nome del campo del *form* e il valore inserito dall'utente per quel campo.
- <<panel>> La GUI Java del *client* dedicato mostra all'utente PDM un pannello contenente informazioni con cui l'utente può interagire.
- <<click>> L'utente PDM interagisce con un pannello della GUI Java facendo *click* su determinati elementi.
- <<popup>> La GUI Java del *client* dedicato mostra all'utente PDM un *popup* dove si chiedono scelte all'utente (ad es. OK, Annulla) o si mostrano eventuali errori.
- <<RMI>> Invocazione di un metodo di un oggetto remoto che risiede sul server fisico del museo chiamato all'interno della GUI Java del *client dedicato* che risiede invece sui *personal computer* degli utenti PDM, tramite la tecnologia RMI.

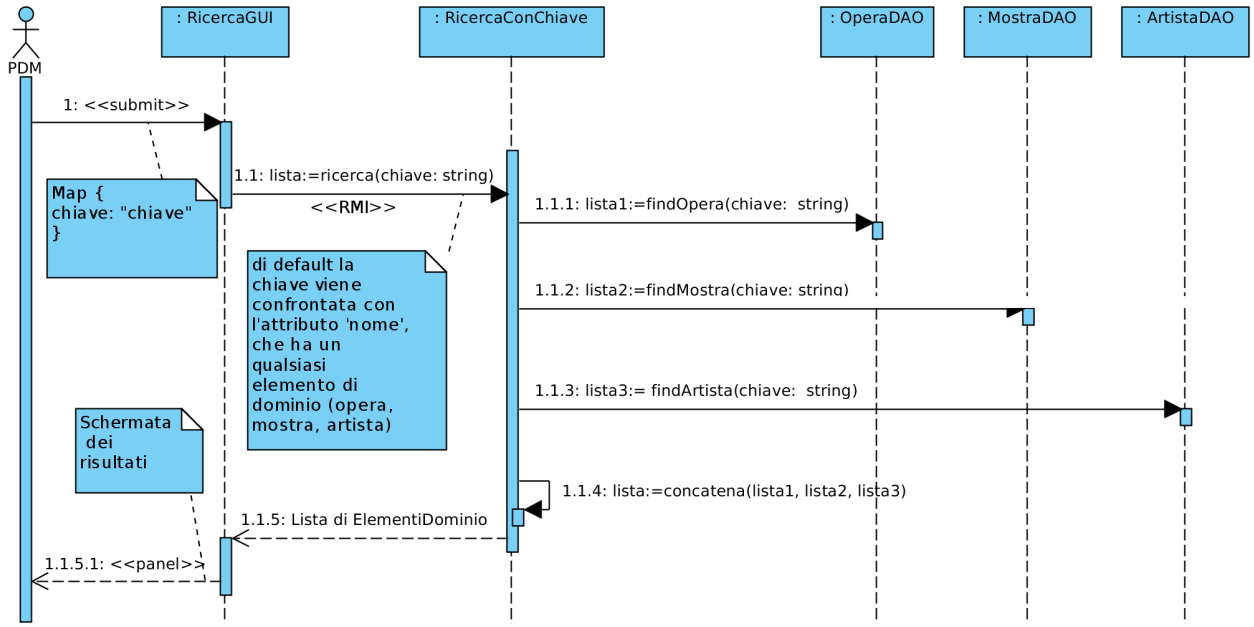
4.1.1.1 SEQ_D_01[UC_01_01]



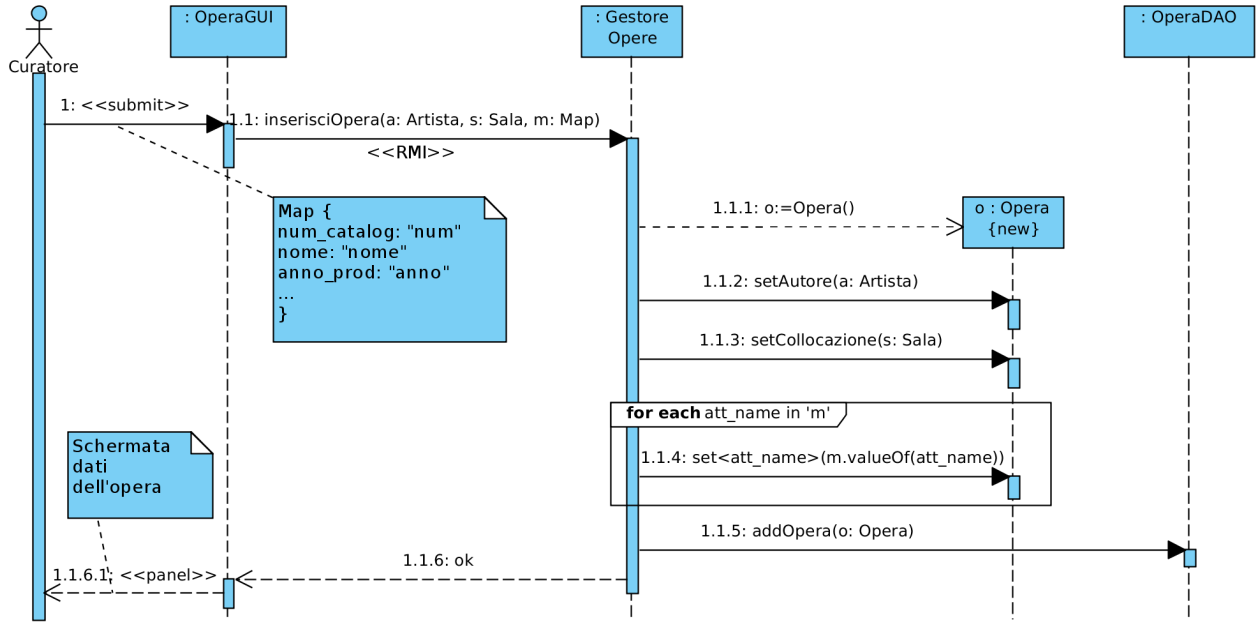
4.1.1.2 SEQ_D_02[UC_01_02]



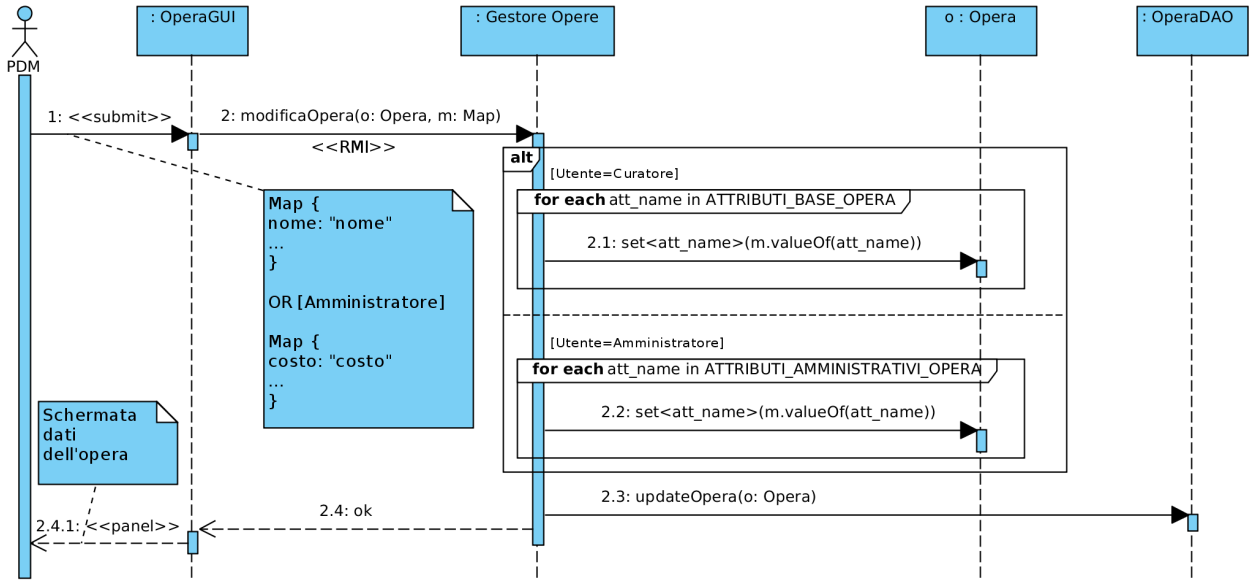
4.1.1.3 SEQ_D_03[UC_02_01]



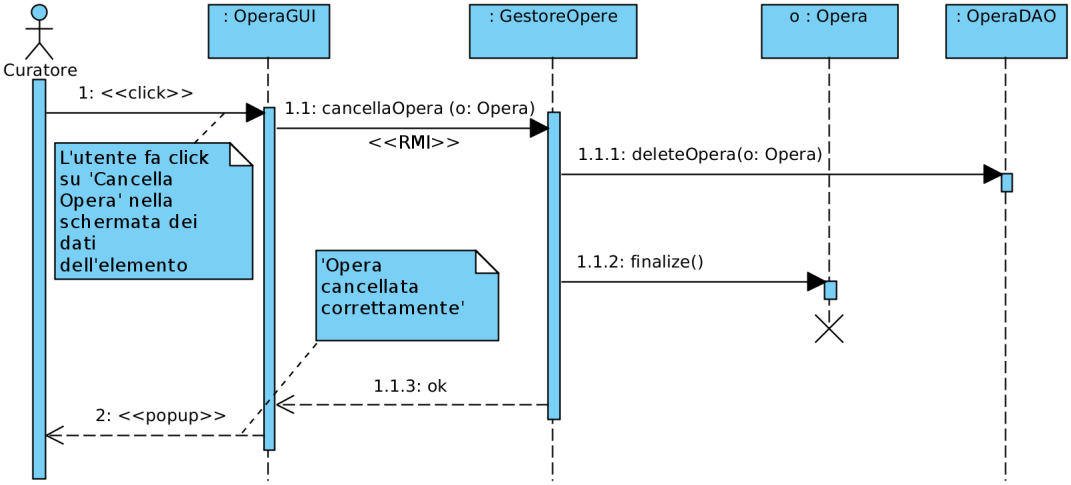
4.1.1.4 SEQ_D_04[UC_04_01_01]



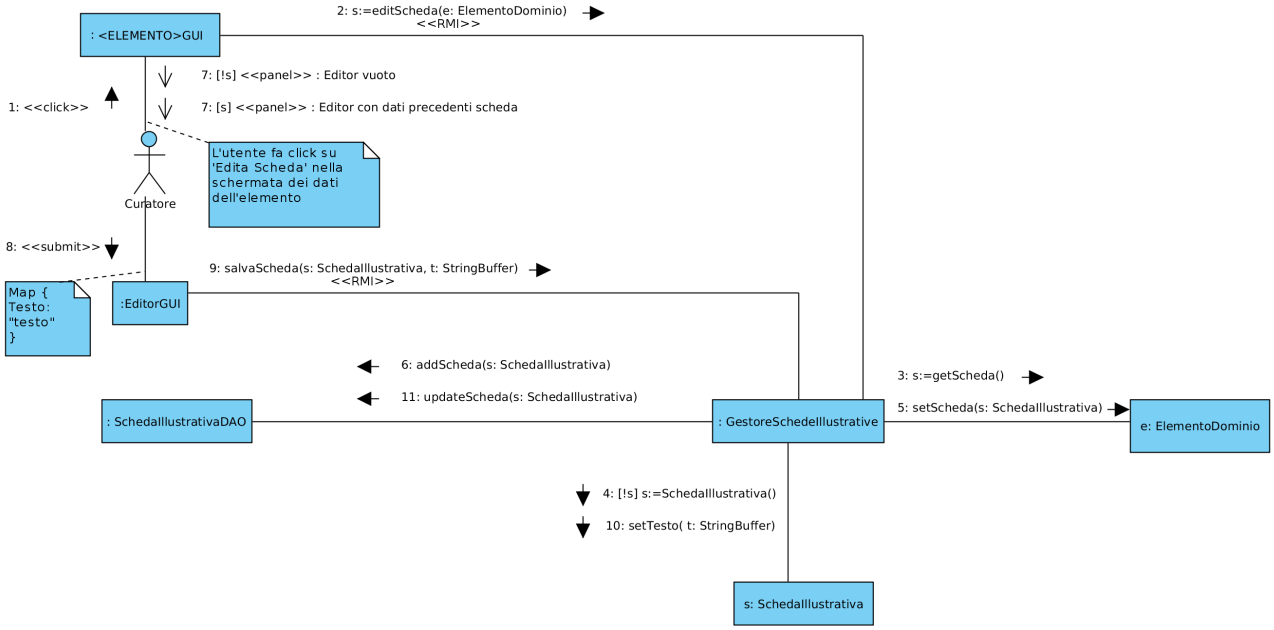
4.1.1.5 SEQ_D_05[UC_04_01_02, UC_05_03]



4.1.1.6 SEQ_D_06[UC_04_01_03]



4.1.1.7 CO_D_01 [UC_04_05_01, UC_04_05_02]

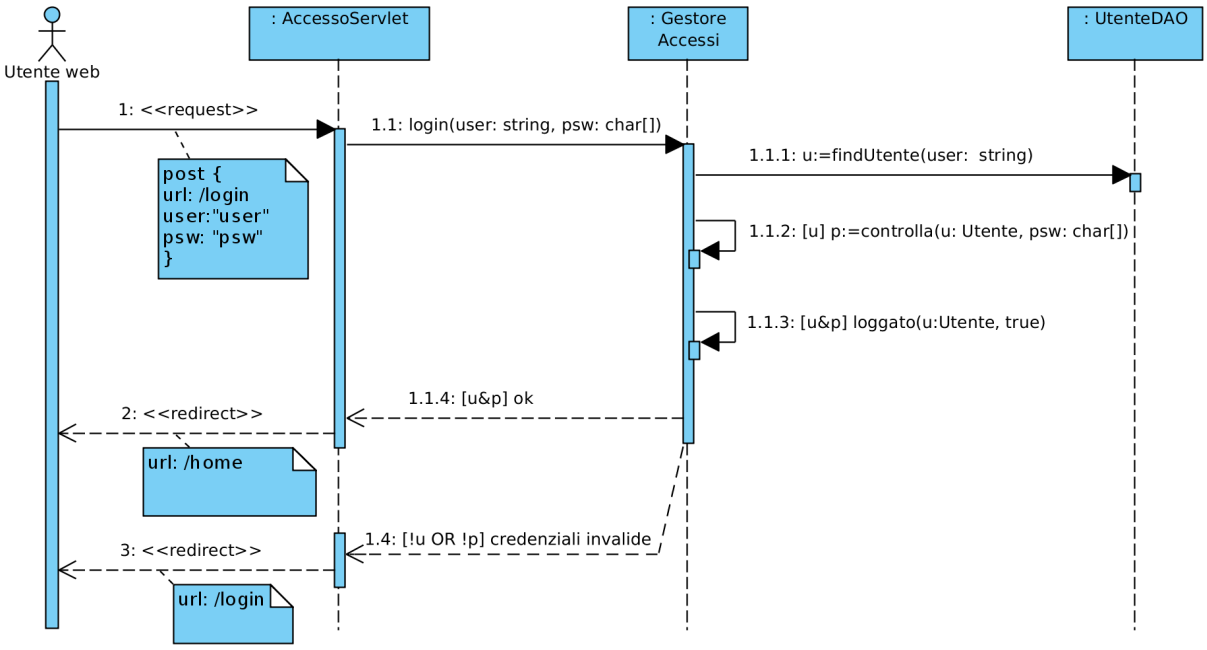


4.1.2 WEB

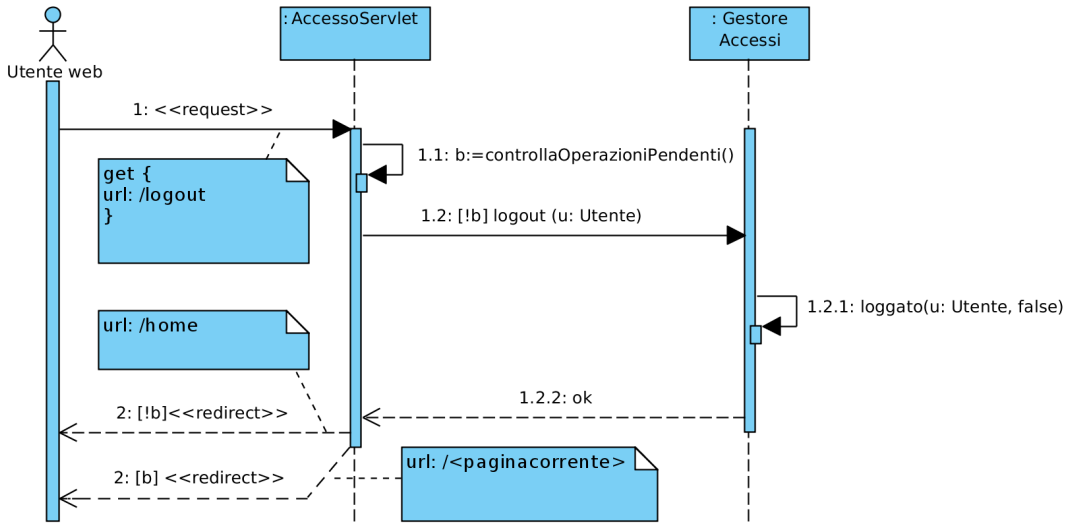
Stereotipi utilizzati:

- <<request>> L'utente web effettua una richiesta http (di tipo *get* o *post*) al web server del museo tramite il proprio browser.
- <<redirect>> Il web server del museo reindirizza l'utente web su una determinata pagina.

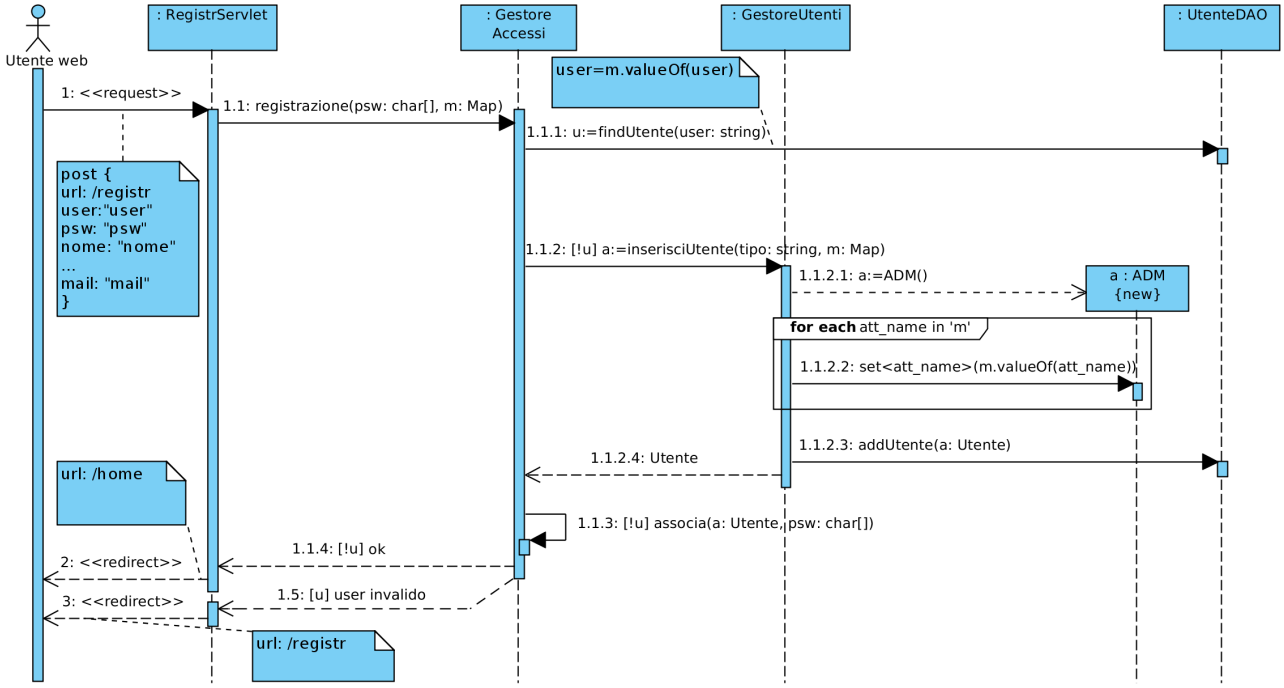
4.1.2.1 SEQ_D_07[UC_01_01]



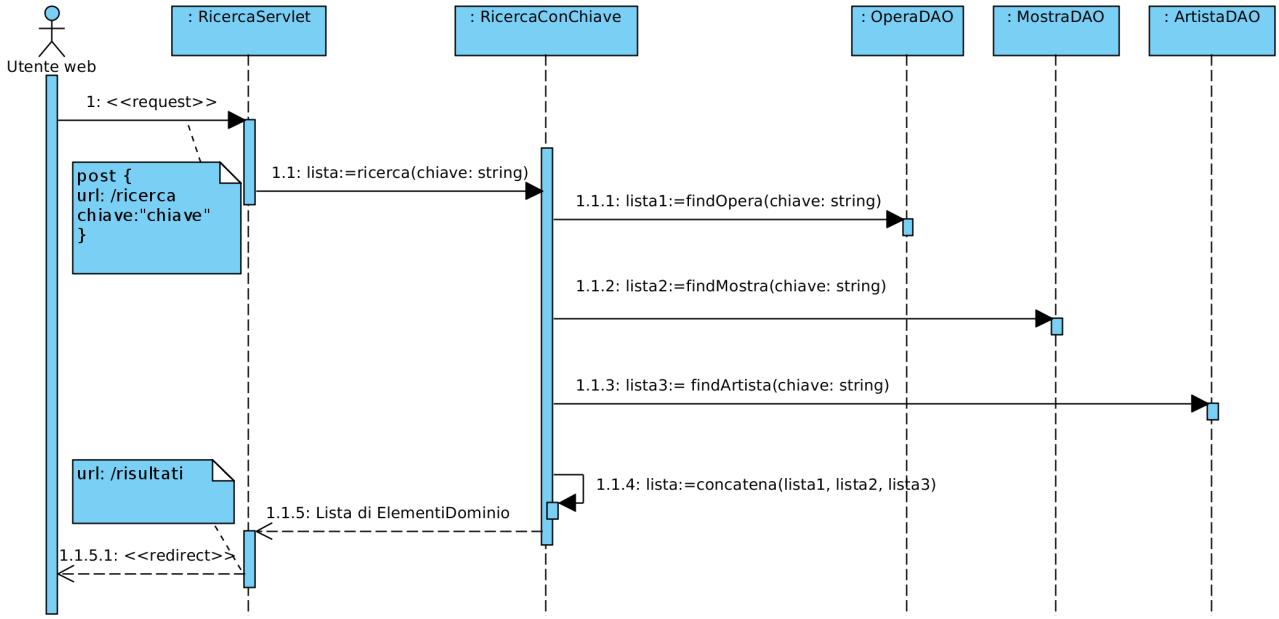
4.1.2.2 SEQ_D_08[UC_01_02]



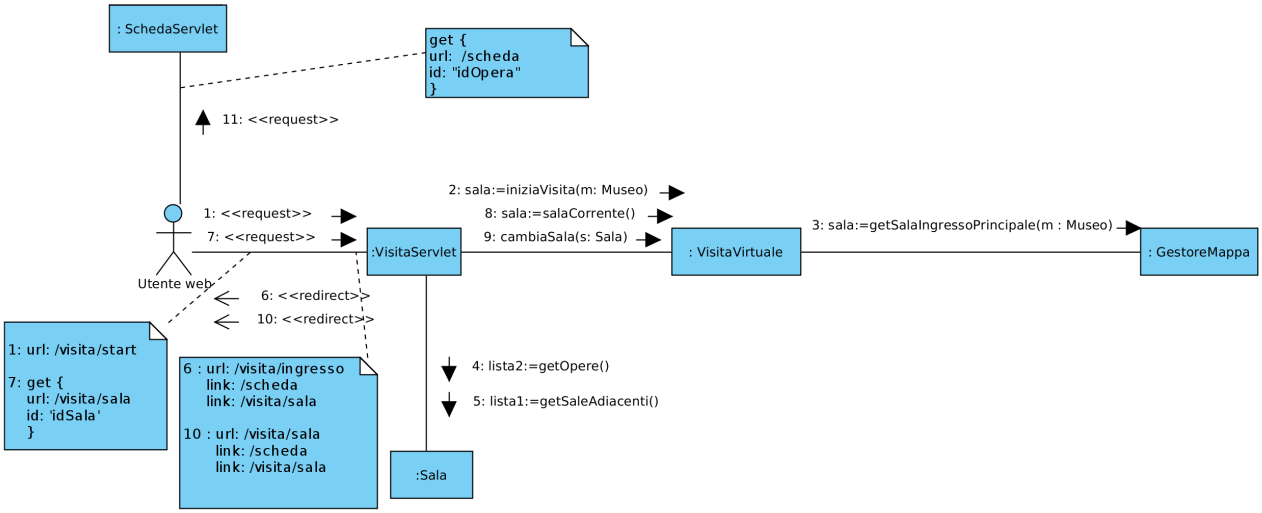
4.1.2.3 SEQ_D_09[UC_01_03]



4.1.2.4 SEQ_D_10[UC_02_01]



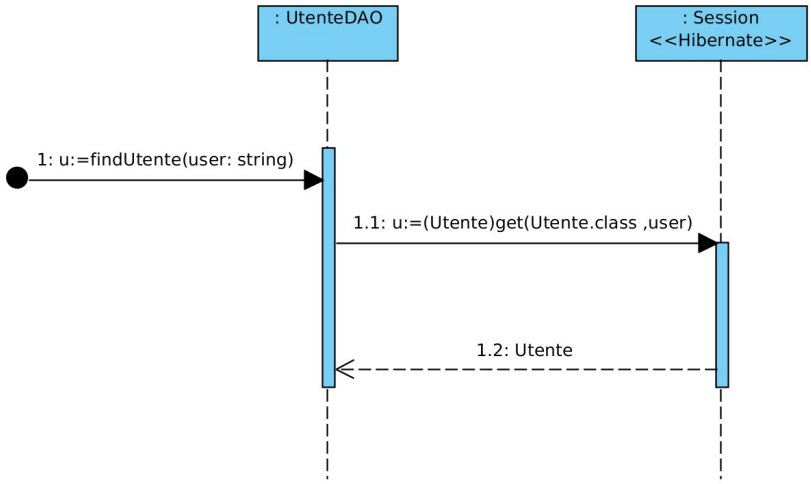
4.1.2.5 CO_D_02[UC_06_04]



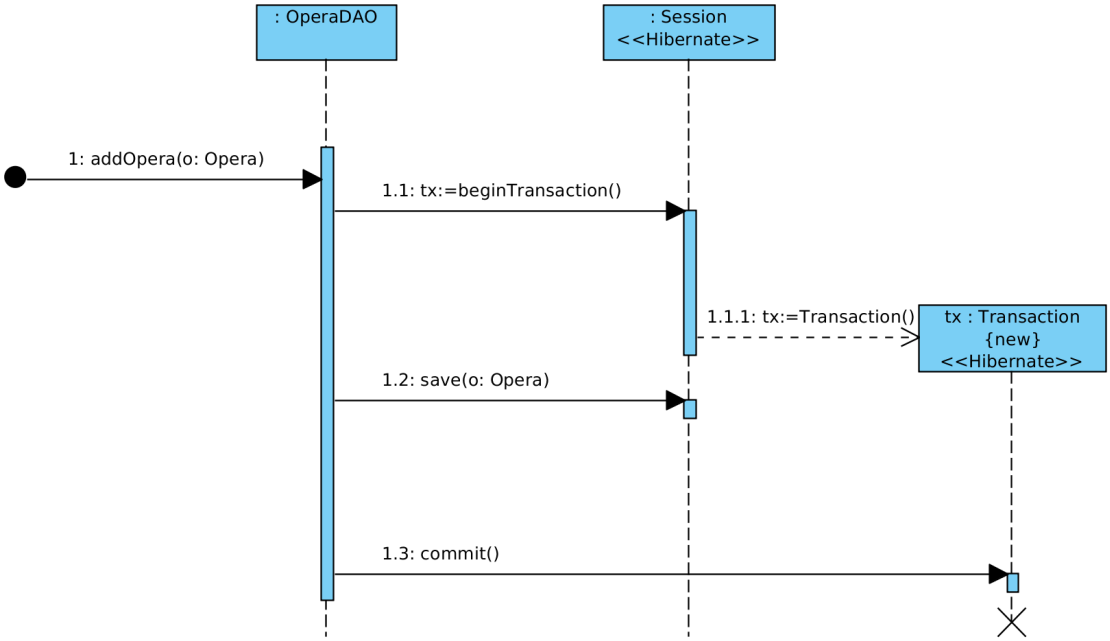
4.1.3 Esempi di comunicazione tra oggetti DAO ed Hibernate

Si presuppone che gli oggetti DAO abbiano accesso a tutti i contenuti pubblici del package org.hibernate e che dispongano di file XML che definiscono la mappatura delle classi java a cui si riferiscono sulle tabelle del database (file ORM) che utilizzerà hibernate previa configurazione.

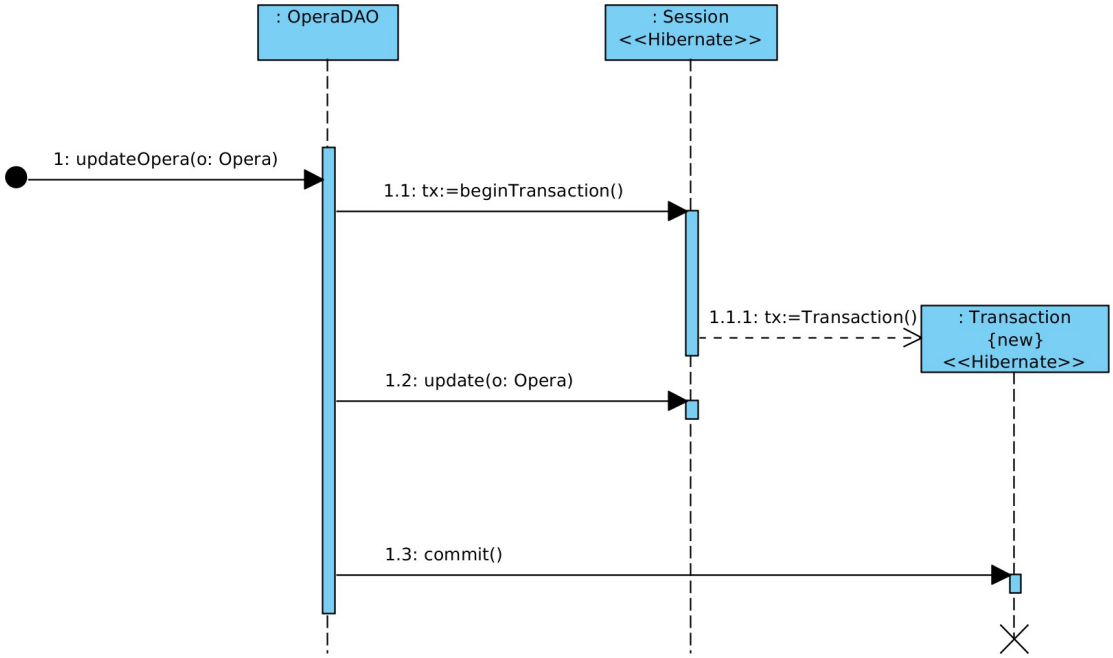
4.1.3.1 SEQ_D_11



4.1.3.2 SEQ_D_12



4.1.3.3 SEQ_D_13



4.1.3.4 SEQ_D_14