Progetto Ingegneria del Software

Documento di Design

"La più vasta raccolta di musica, sempre a tua disposizione"

Gianluca Amato 1691171
Patrizio Conte 1618639

Indice

1. Prefazione

Lo scopo di questo documento è di evolvere e approfondire il modello di analisi aggiungendo dettagli relativi all'implementazione. Verrà mostrata una specifica architetturale del sistema, e aggiungeremo il necessario per rendere possibile l'implementazione dei requisiti.

2. Architettura del sistema

L'architettura software (anche chiamata architettura logica) del sistema è di tipo client/server multistrato, che indica un'architettura di rete nella quale un *client* (computer, cellulari, ecc) si connette ad un *server* per la fruizione di un certo servizio, appoggiandosi alla sottostante architettura protocollare. In particolare:

- il lato server è formato da:
  - *application server*: che implementa la logica dell'applicazione;
  - *web server*: che ascolta le richieste provenenti dalla rete;
  - *content delivery network (CDN)*: usato per distribuire i contenuti multimediali, nel nostro caso i brani;
  - *database manager*: che gestisce l'accesso al database.
- invece il lato client è composto da terminali anche detti *client devices* che comunicano con il web server tramite un browser attraverso protocolli di rete come HTTP e HTTPS.

Per il web server si è scelto di utilizzare il software open source **NGINX** [Engine-ex]; la scelta è ricaduta su questo software, dopo attenti studi, poichè risulta molto veloce nel soddisfare le richieste dei client non perdendo in prestazioni e siccome si prevede un grande carico di lavoro questa caratteristica è molto utile per la nostra applicazione. Inoltre permette un adeguato bilanciamento del carico di lavoro in base alle richieste ricevute e una minore propenzione al "crash" rispetto ad altri web server [fonti ufficiali].

La content delivery network (CDN) è una rete per la distribuzione dei contenuti estremamente distribuita per la delivery di contenuti multimediali, nel nostro caso i brani. Questa rete è disseminata in moltissime ubicazioni fisiche e di rete, in modo da poter rispondere direttamente alle richieste degli utenti finali, quindi questa rete funge da intermediario tra un server di contenuti e i relativi utenti finali o client. La CDN scelta per questo progetto è **Cloudflare** con il piano "Enterprise" per le sue alte prestazioni fornite e per l'alta distribuzione di server in tutto il globo.

Per il database manager, si è scelto di utilizzare **PostgreSQL 12.0**; sia per la sua licenza open source, sia per la molteplicità di funzioni permesse indispensabili per *Spotify*. Per la comunicazione tra applicazione e database si useranno delle classi *DAO* (Data Access Object) che verranno descritte più avanti in questo documento.

L'applicazione si interfaccerà, tramite protocolli standard di comunicazione (come HTTPS, TCP/IP), a sistemi esterni già esistenti come per esempio l'invio di posta elettronica.

3. Classi di Progettazione

Di seguito troviamo le classi di progettazione, ovvero le classi descritte nel modello di analisi, ridefinite con una descrizione dei dettagli di progettazione per la futura implementazione del sistema. In vista della realizzazione dell'architettura sono state aggiunte classi

- *DAO* per la gestione della persistenza dei dati e quindi per la fornitura di un'interfaccia al database;
- *Servlet* per la generazione di pagine web.

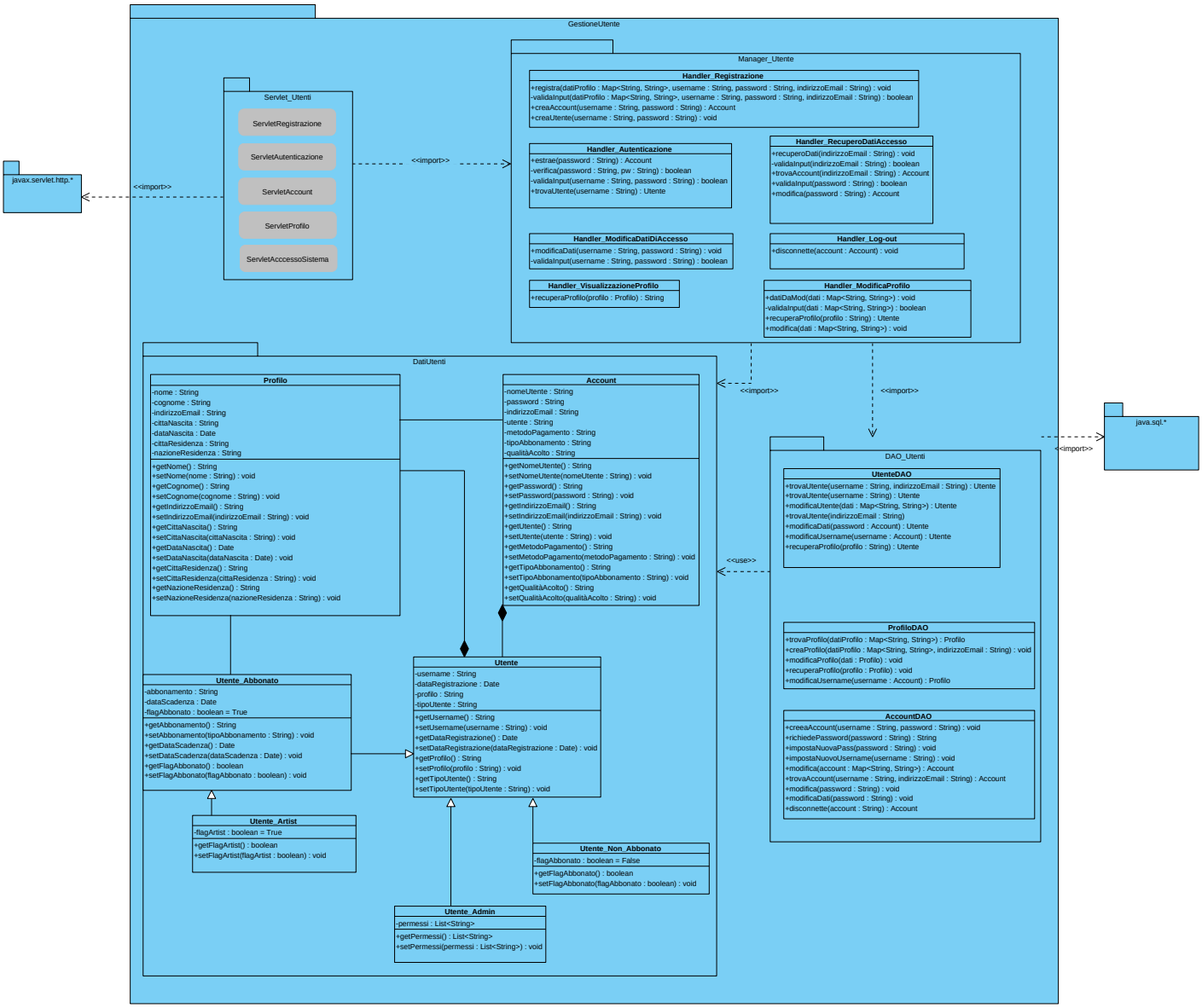
Per semplicità di progettazione si è scelto di usare le Servlet, ovvero programmi scritti in Java residenti sul server in grado di gestire le richieste generate da uno o più client, attraverso uno scambio di messaggi tra il server ed i client stessi che hanno effettuato la richiesta. Il web server da noi scelto (*NGIN-X*) non è in grado di interfacciarsi con le Servlet e per questo motivo si è scelto di usare il web server come un "reverse proxy", che riceve le richieste e le inoltra al server di backend appropriato, a tale scopo si è scelto di utilizzare il server di backend *Apache Tomcat*; mantenendo quindi le alte prestazioni fornite dal web server scelto.
Per rendere i diagrammi e la struttura più facili da leggere abbiamo suddiviso il sistema in più subpackage con funzionalità correllate. Vengono mostrate inoltre le librerie usate, presenti nella documentazione ufficiale del linguaggio; in particolare:

- javax.servlet.http.*, che è un'estensione del framework di base javax.servlet, usata per l'implementazione delle richieste provenienti dai client;
- java.sql.*, fornisce le API per l'accesso e per il recupero dei dati nel database; per l'accesso al database si adotta il pattern *DAO* con la realizzazioni di query parametrizzate.

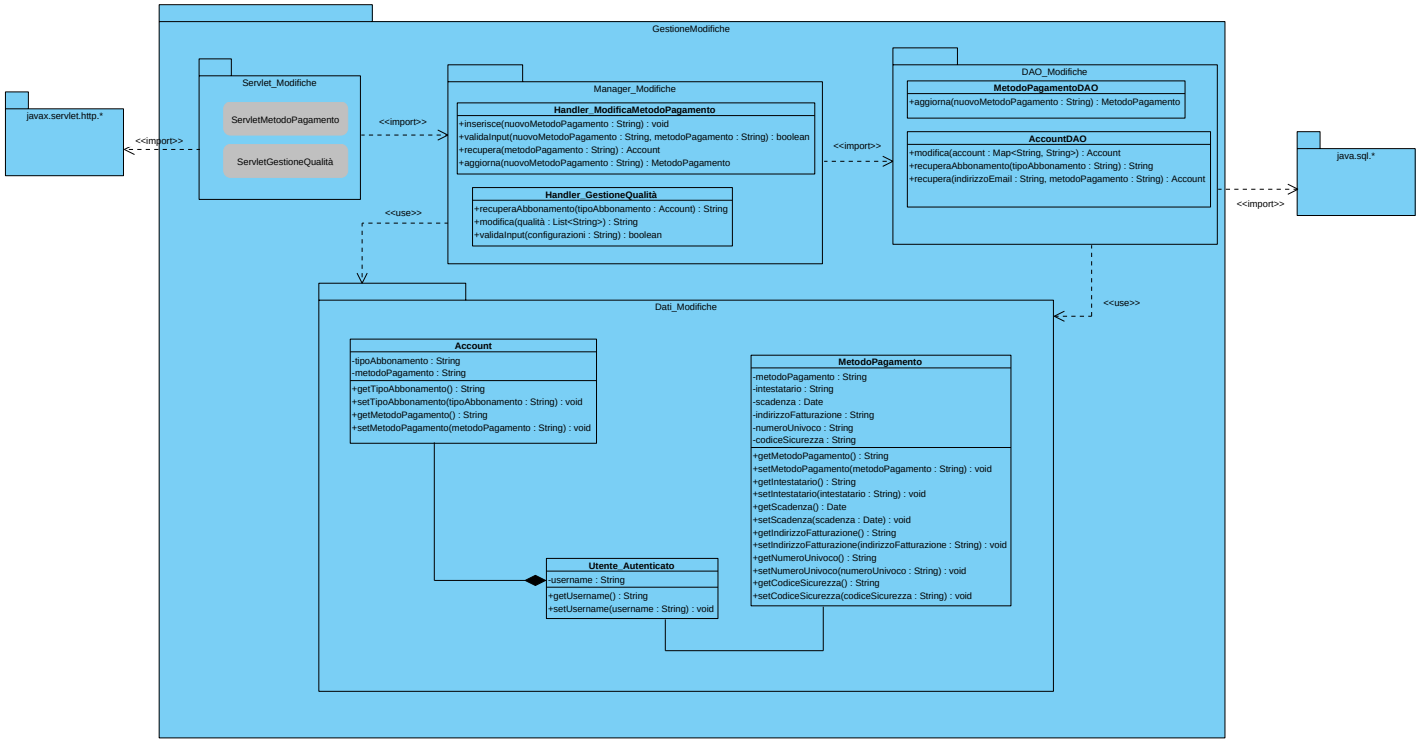
3.1 Classi di progettazione in dettaglio

Di seguito troviamo la suddivisione delle classi nei vari package e le dipendenze fra i package in dettaglio.

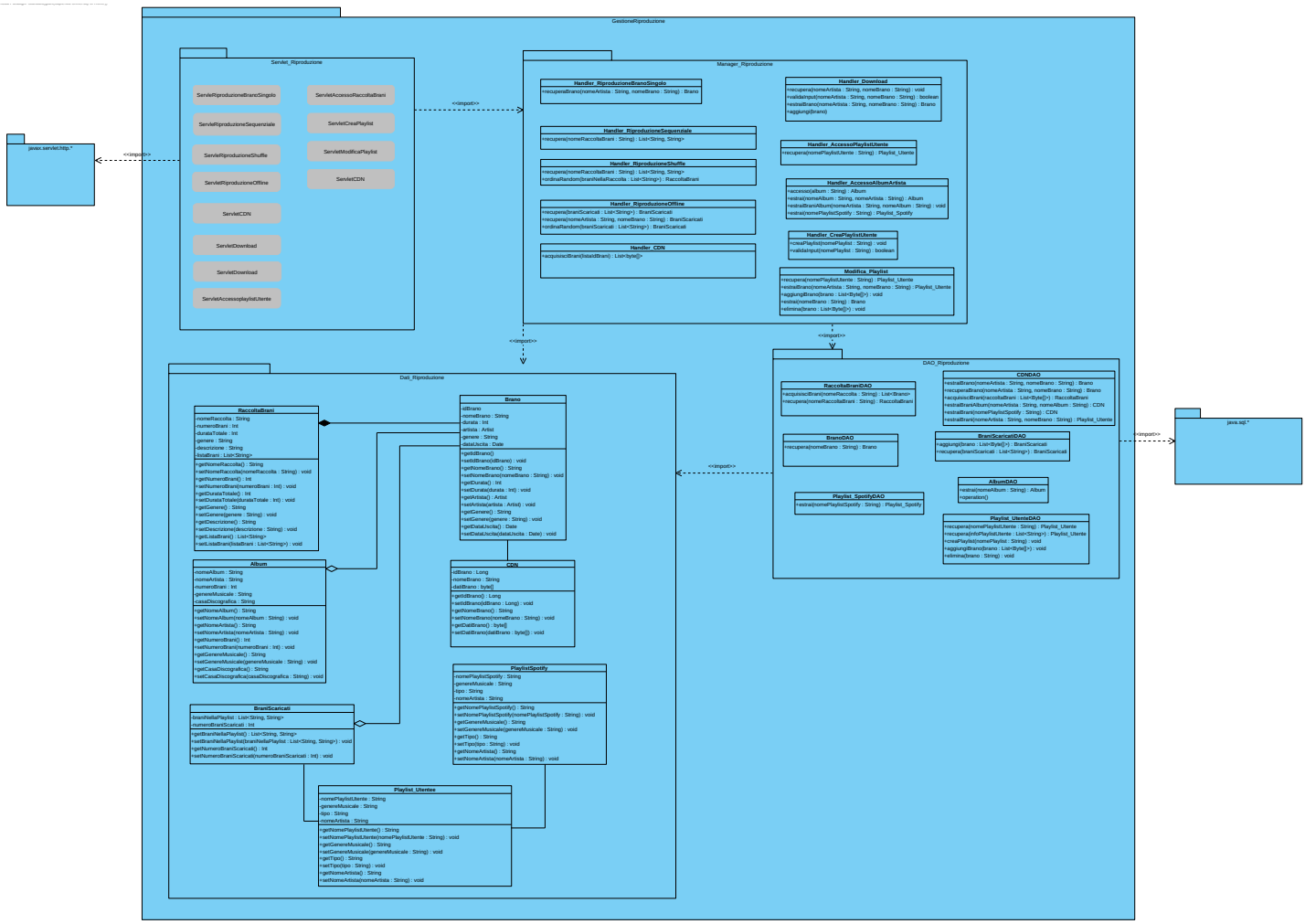
3.1.1 Package Gestione Utenti



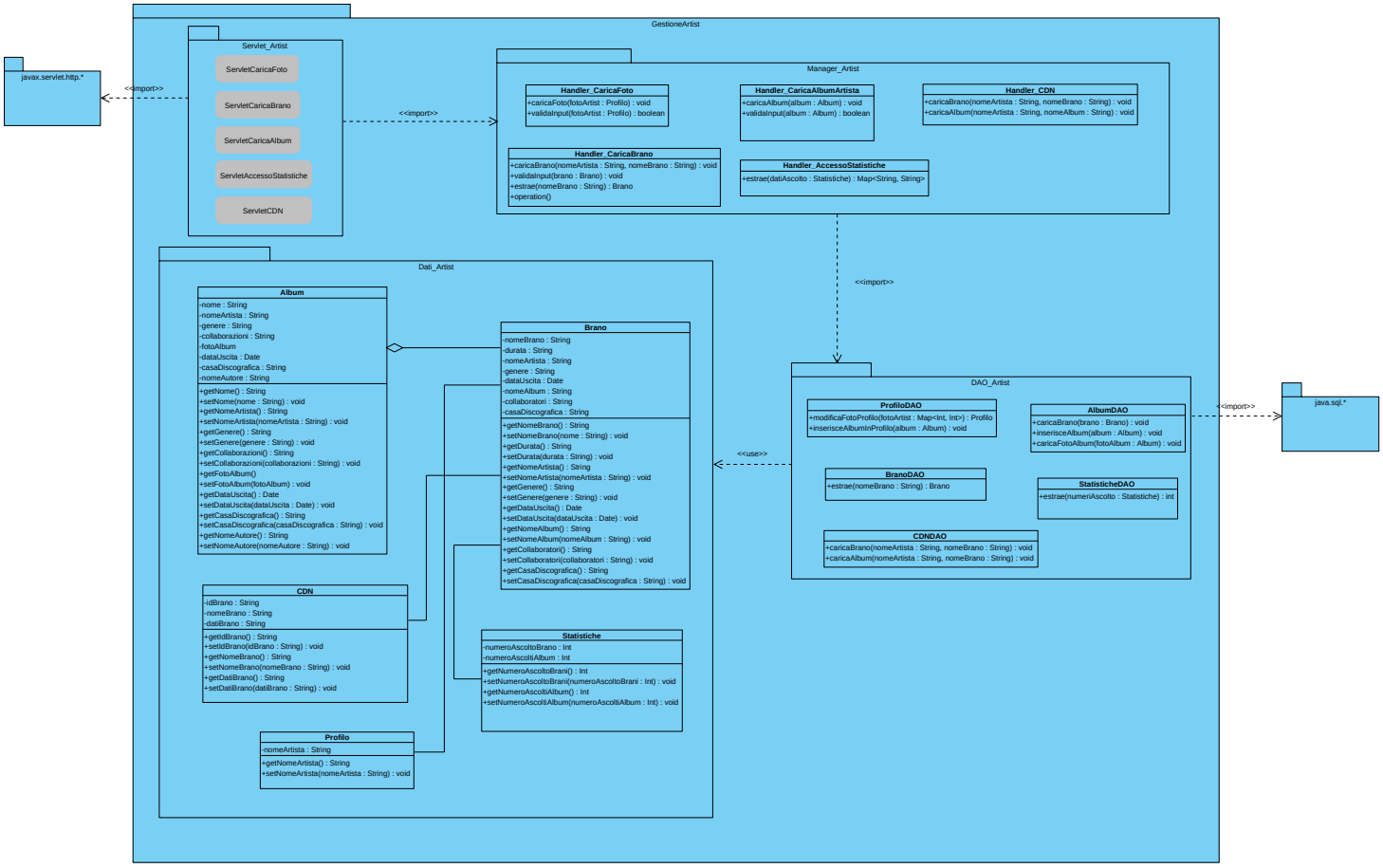
3.1.2 Package Gestione Modifiche



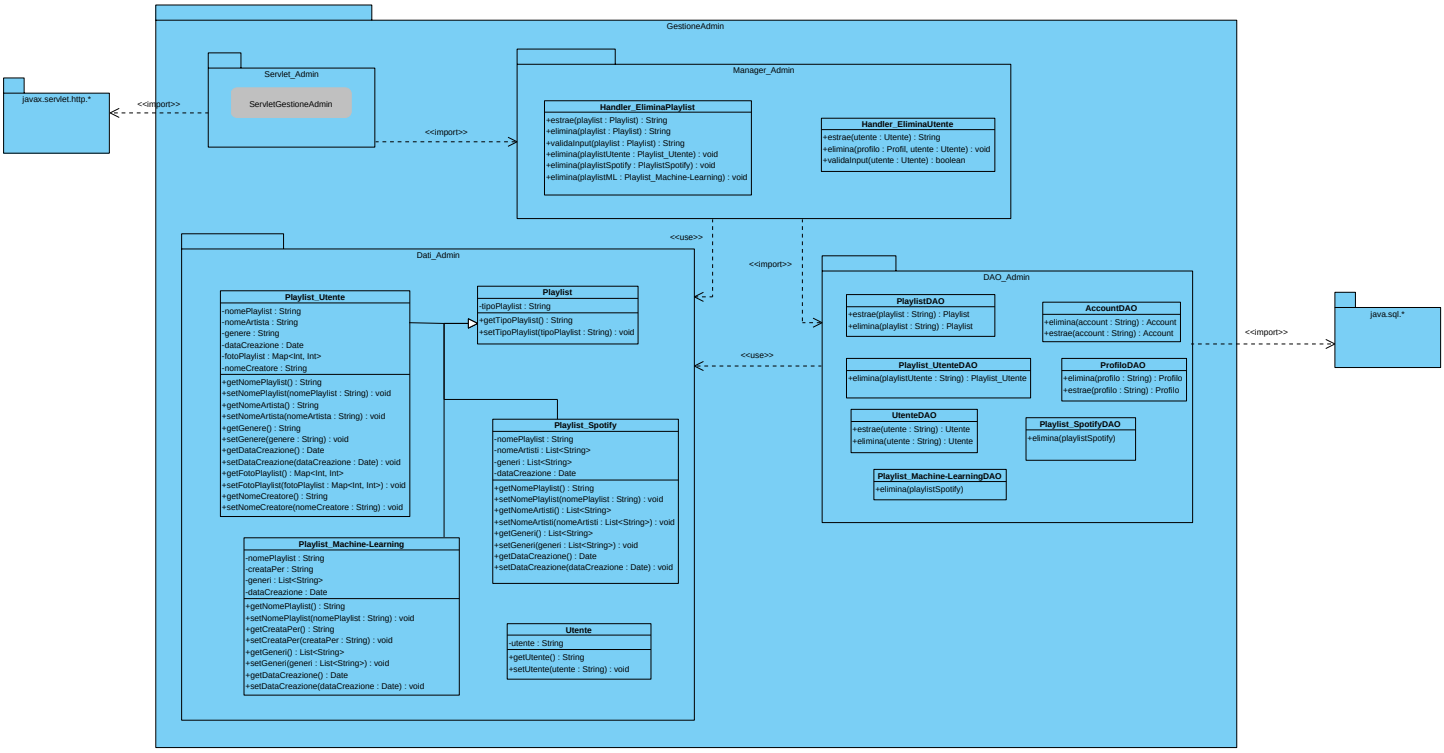
3.1.3 Package Gestione Riproduzione



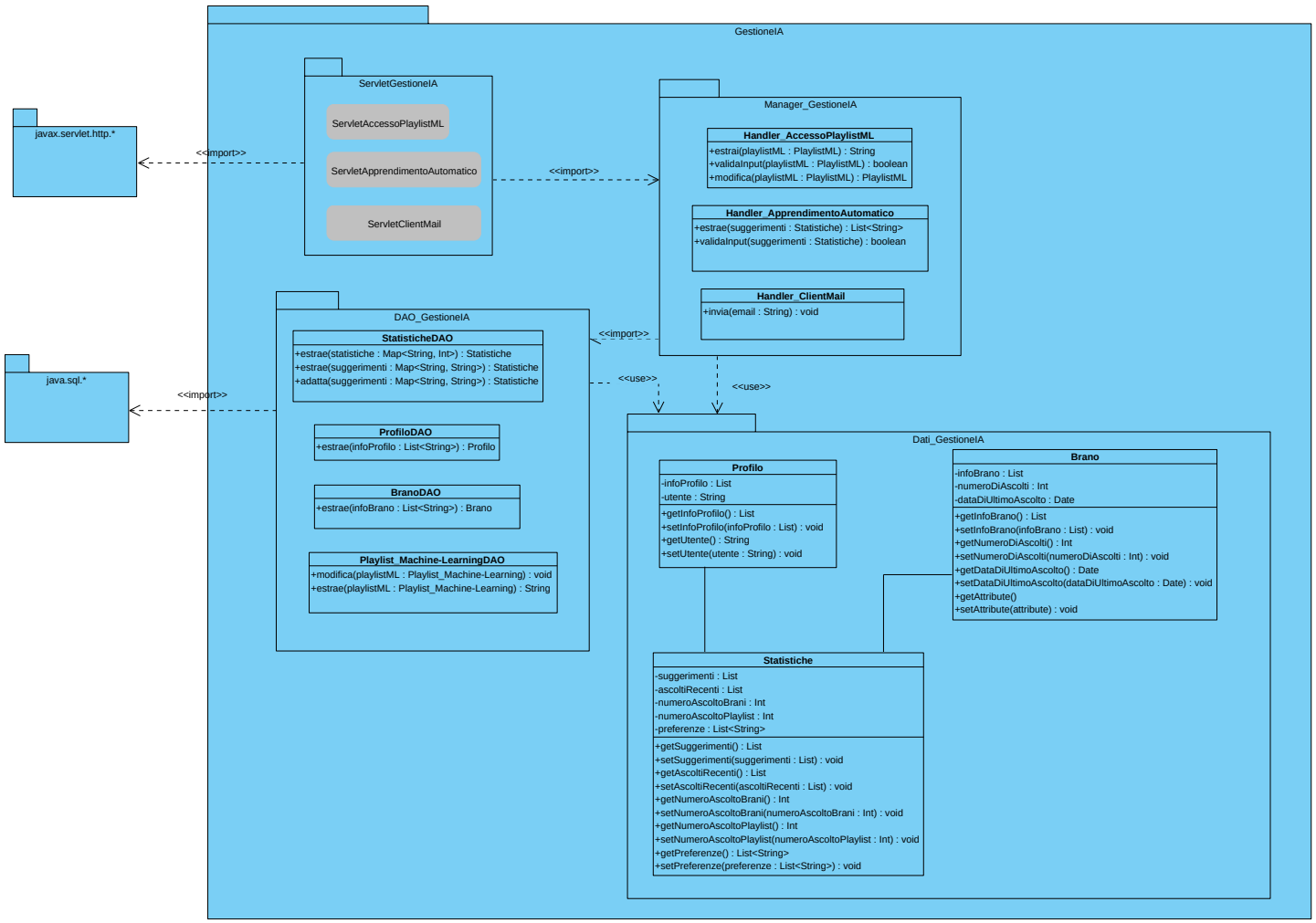
3.1.4 Package Gestione Artist



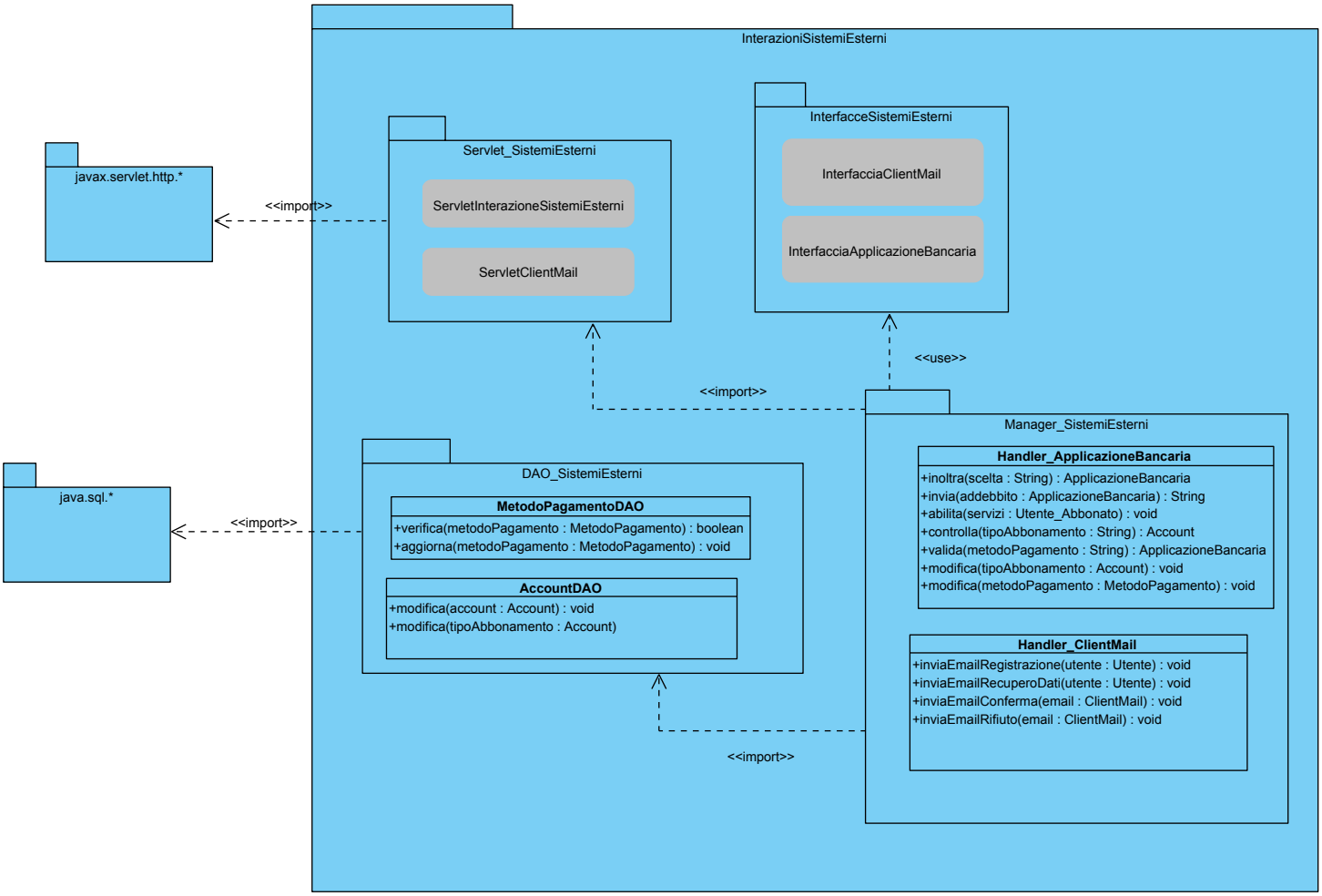
3.1.5 Package Gestione Admin



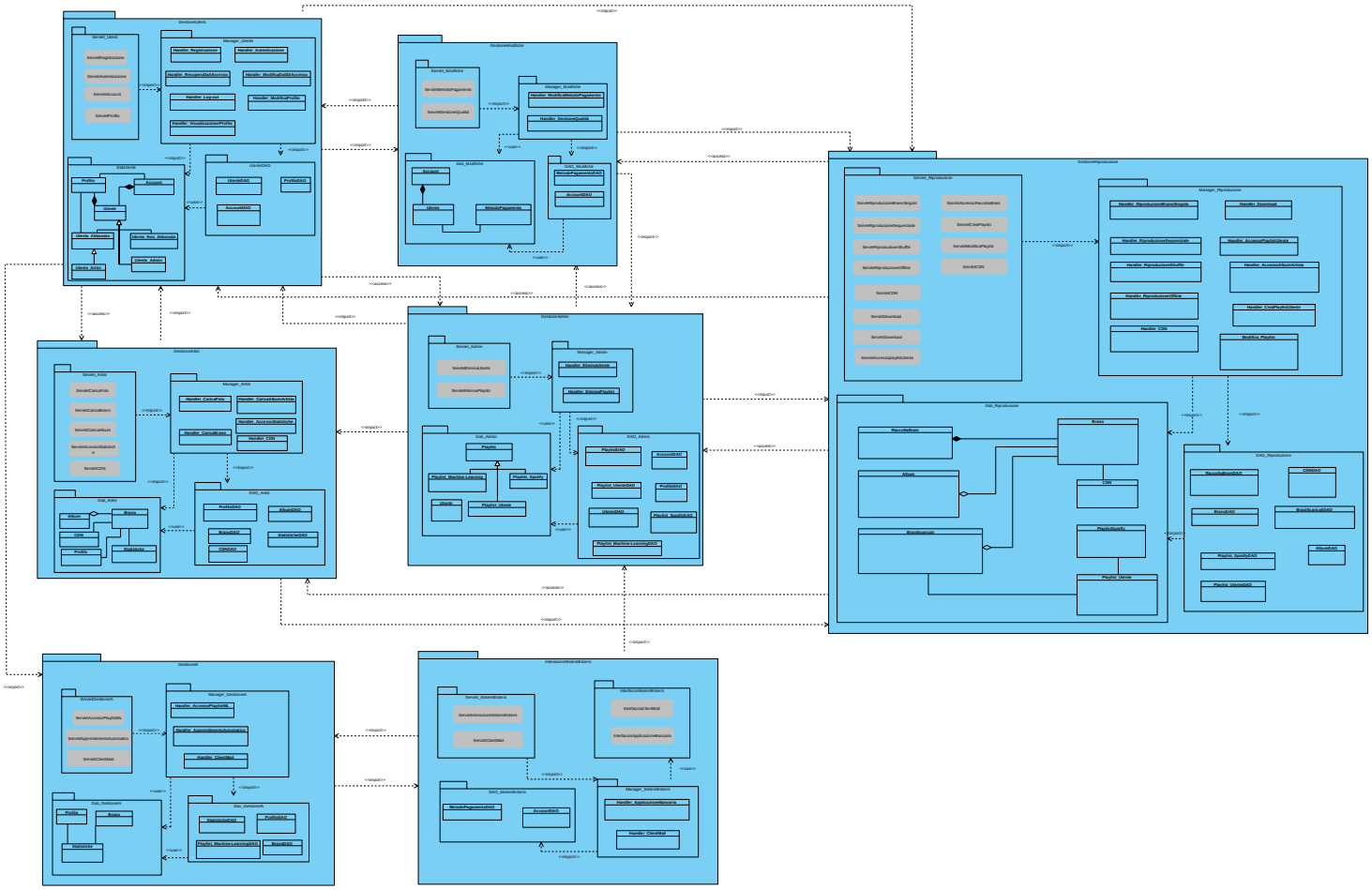
3.1.6 Package Gestione IA



3.1.7 Package Interazioni Sistemi Esterni



3.1.8 Diagramma contentente le dipendenze tra i vari package



4. Diagrammi di Sequenza

In questo capitolo si illustrano i diagrammi di sequenza per completare la fase di analisi già descritta nel documento di Analisi. I diagrammi di sequenza che troviamo di seguito si differenziano da quelli del documento di analisi per il livello di dettaglio fornito; nei diagrammi di sequenza di questo documento si illustrano inoltre i valori che ogni operazione/classe deve restituire.

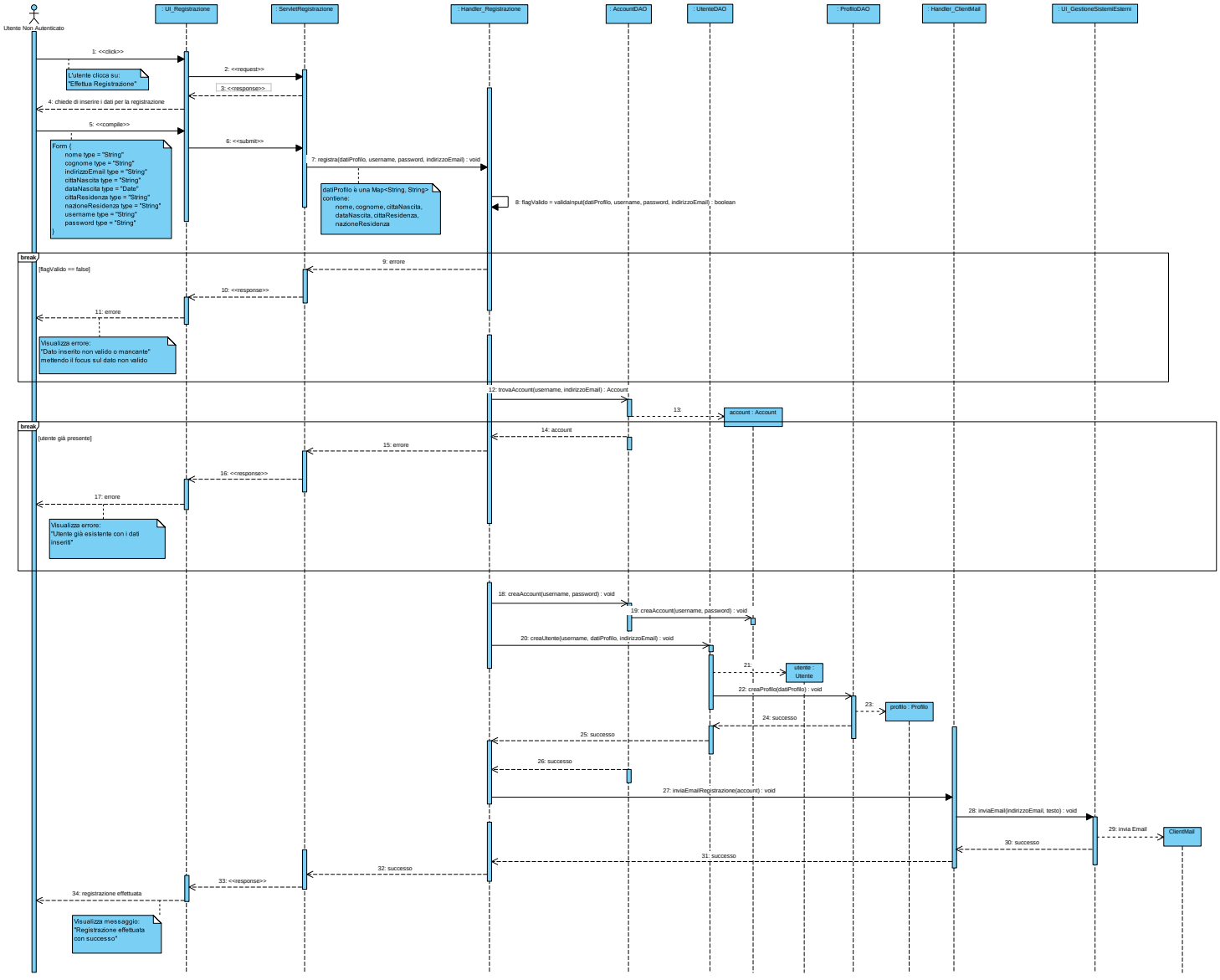
Nei diagrammi sono stati inseriti alcuni stereotipi per facilitare la lettura e la comprensione delle comunicazioni con le Servlet, di seguito troviamo una spiegazione più dettagliata di ogni stereotipo usato.
Stereotipi:

- <<click>>: rappresenta il click dell'utente;
- <<request>>: rappresenta una generica richiesta inviata dall'utente alla Servlet;
- <<compile>>: rappresenta la compilazione di un form da parte dell'utente;
- <<response>>: rappresenta la risposta della Servlet;
- <<submit>>: rappresenta la sottomissione di un form ad una Servlet.

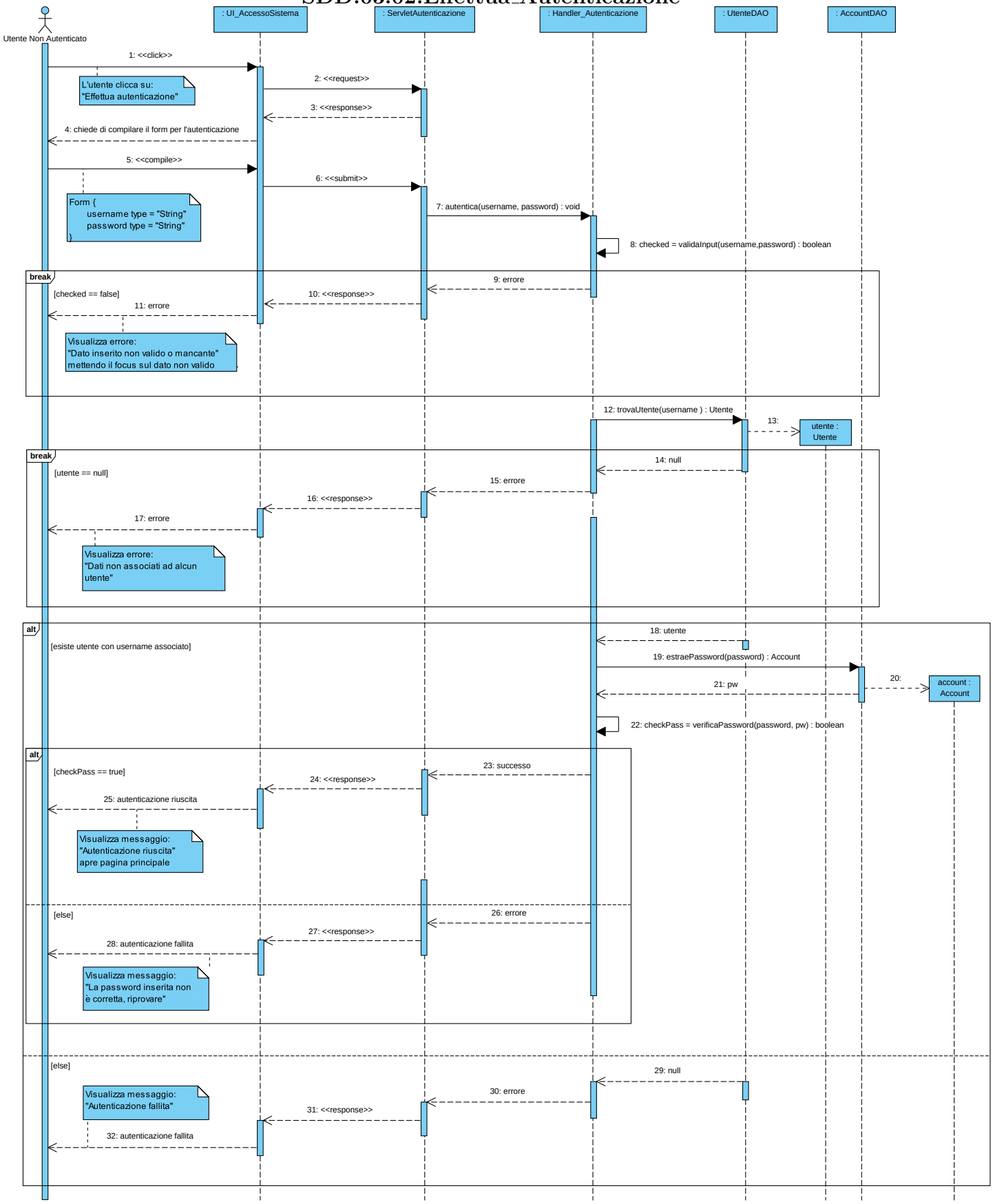
4.1 Diagrammi di sequenza in dettaglio

4.1.1 Sottosistema Gestione Accesso

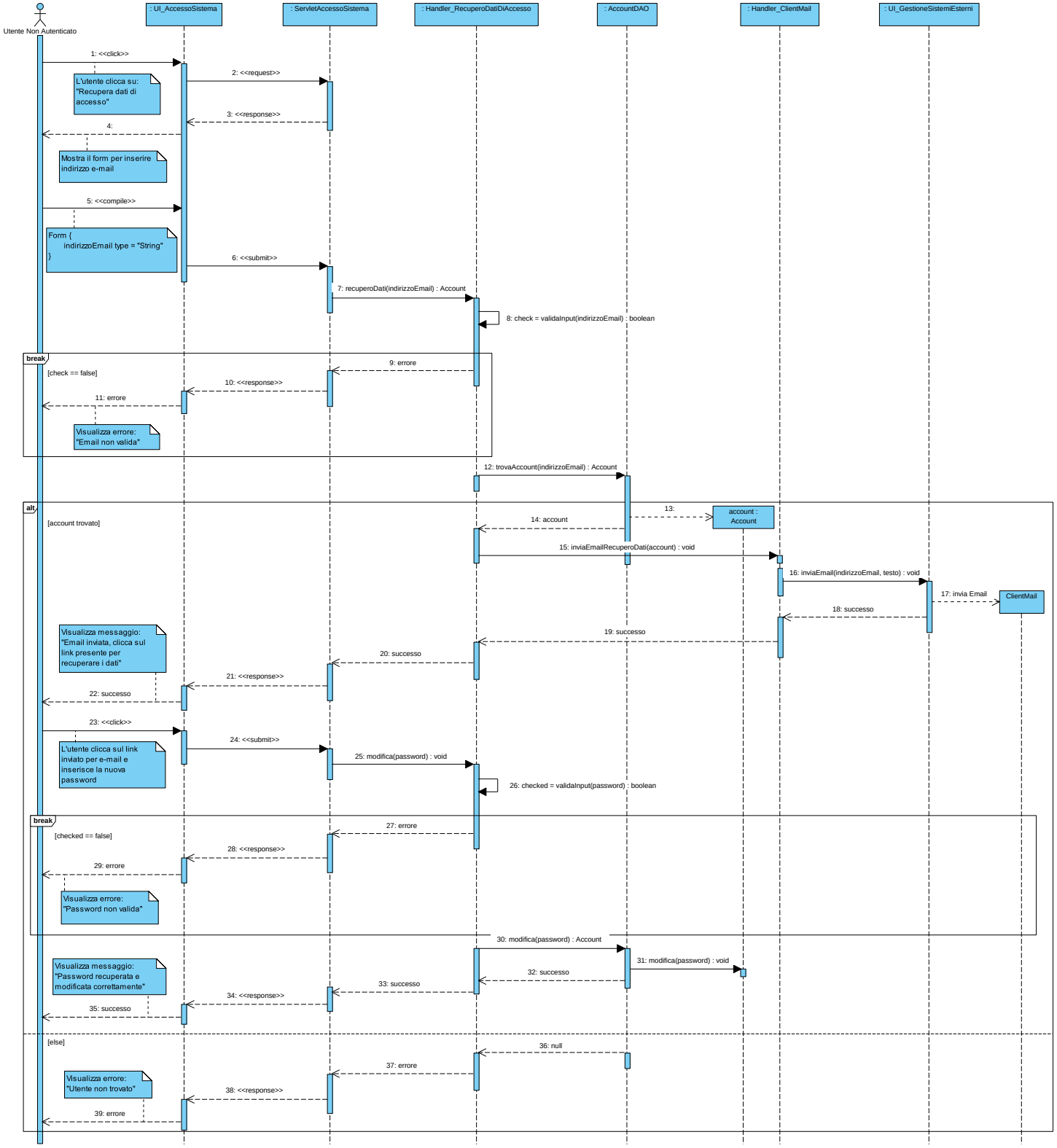
SDD.01.01.Effettua_Registrazione



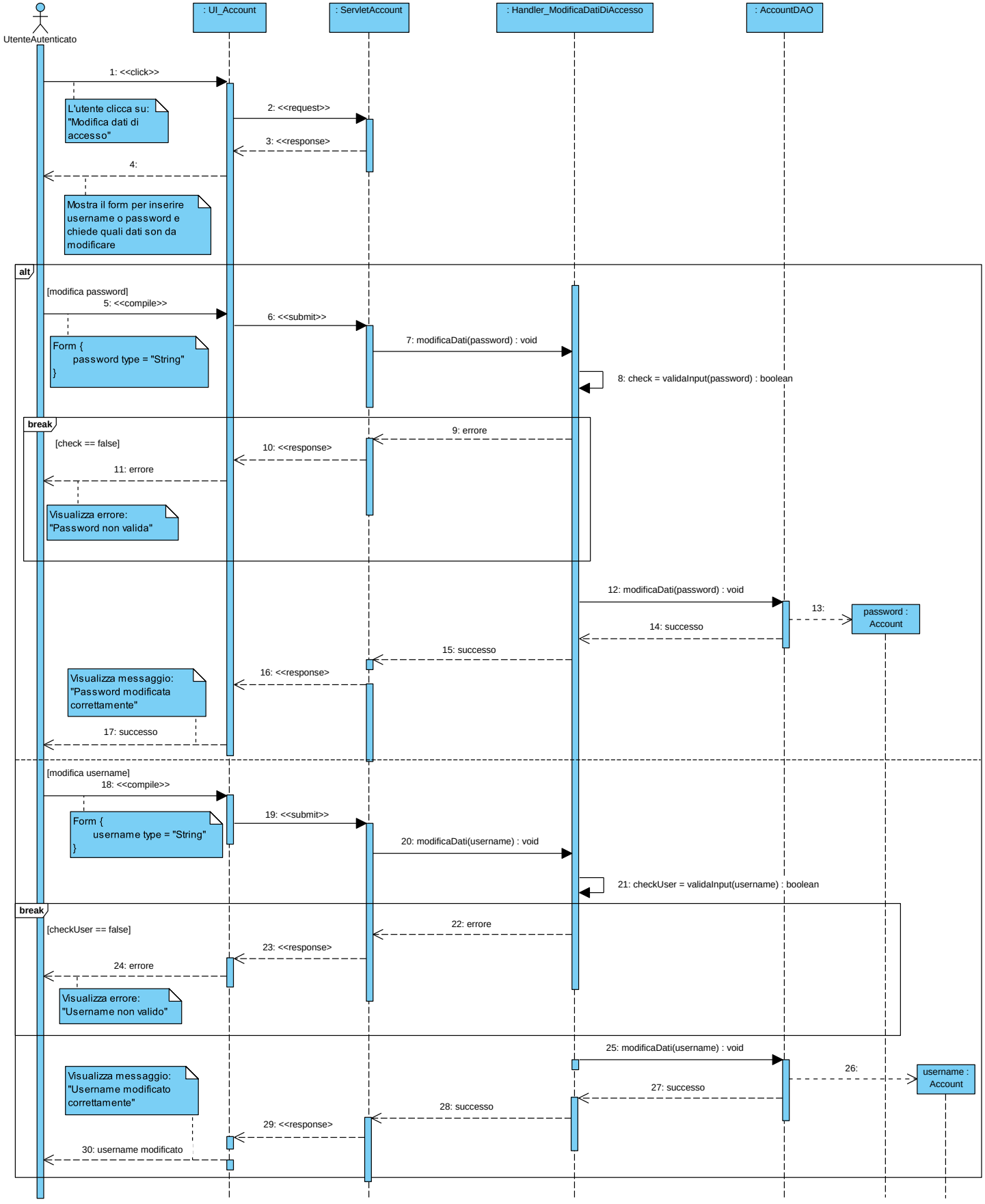
SDD.03.02.Effettua_Autenticazione



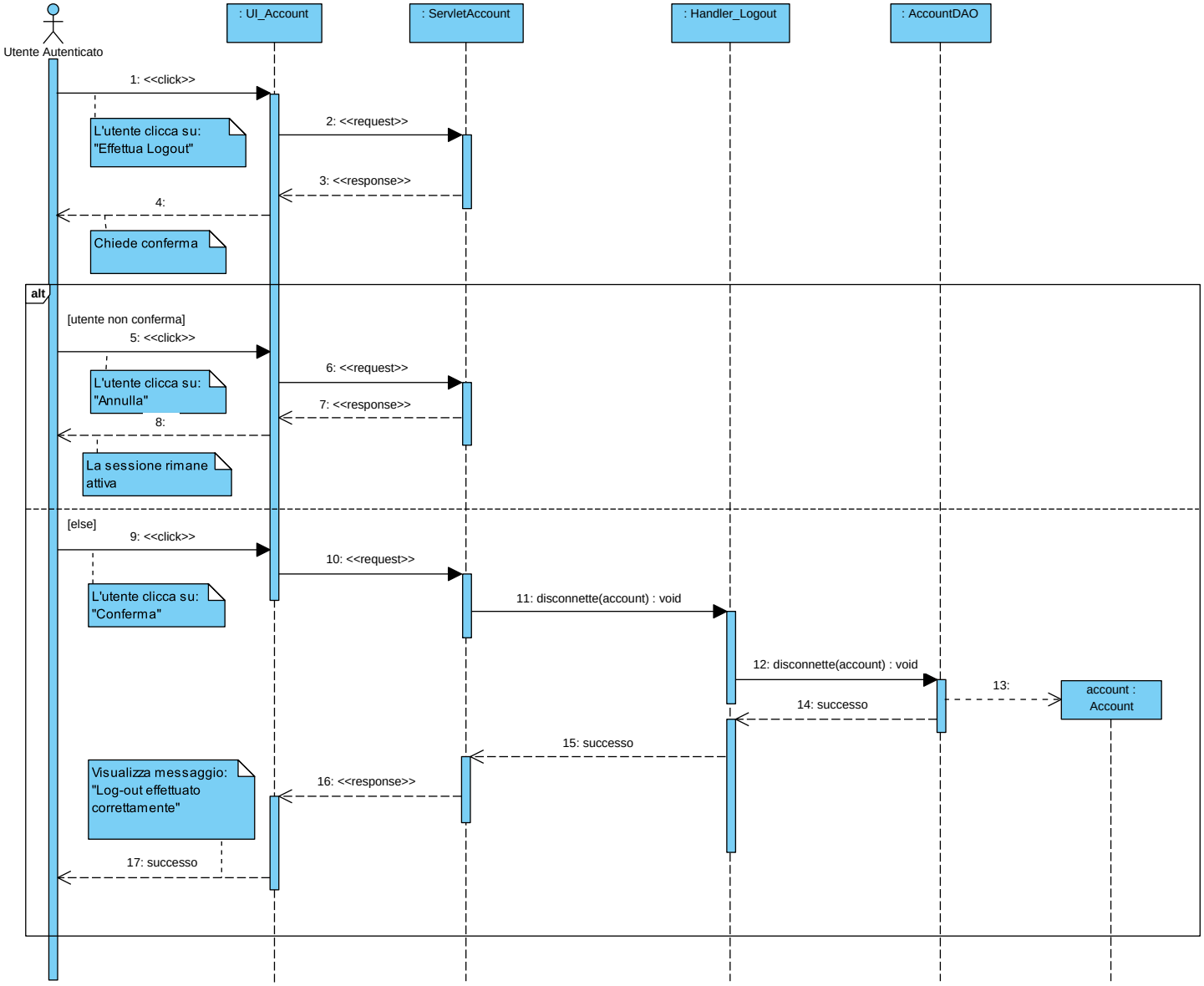
SDD.02.01.01.RecuperaDatiDiAccesso



SDD.04.02.01.ModificaDatiDiAccesso

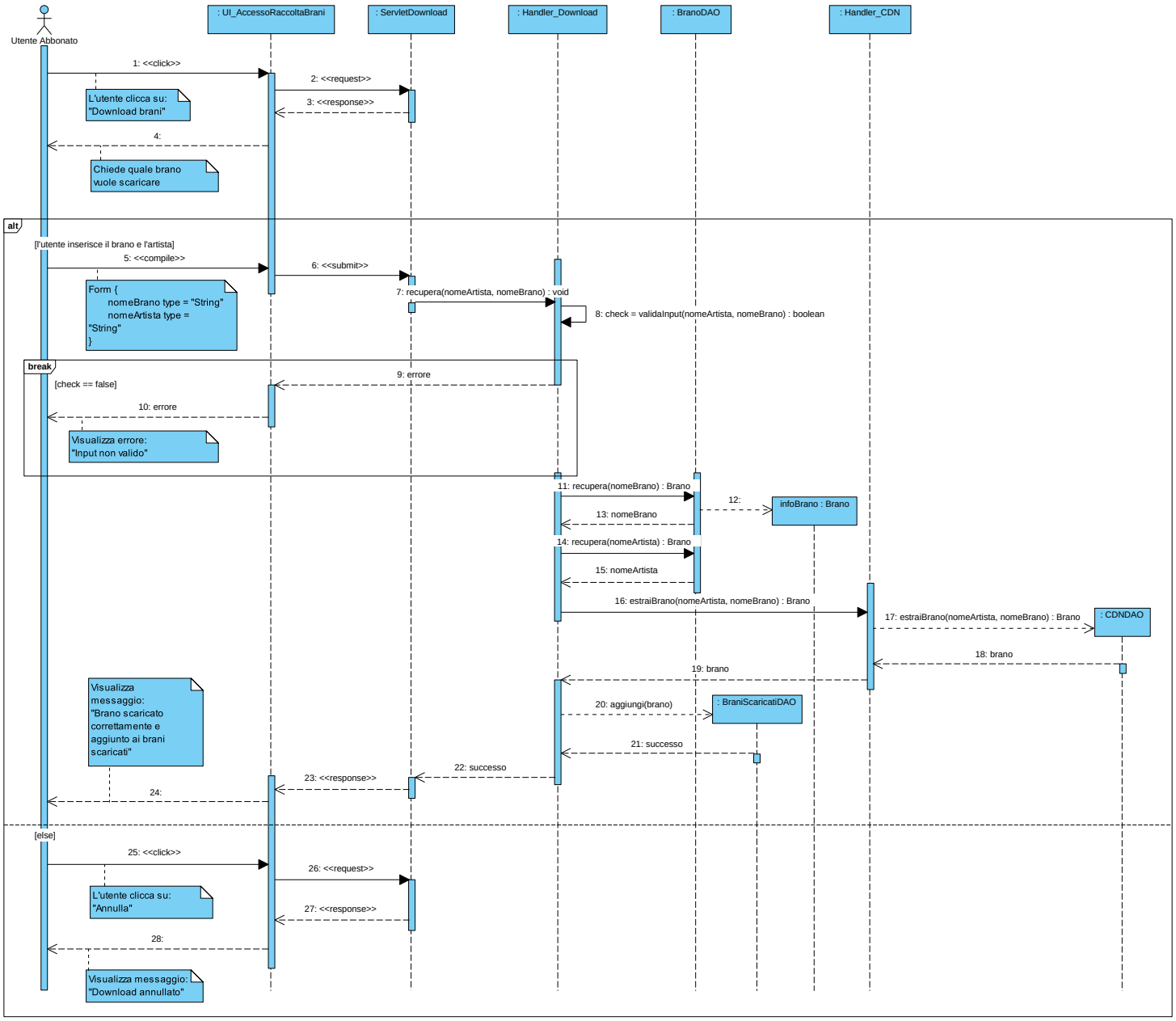


SDD.05.02.02.EffettuaLog-out

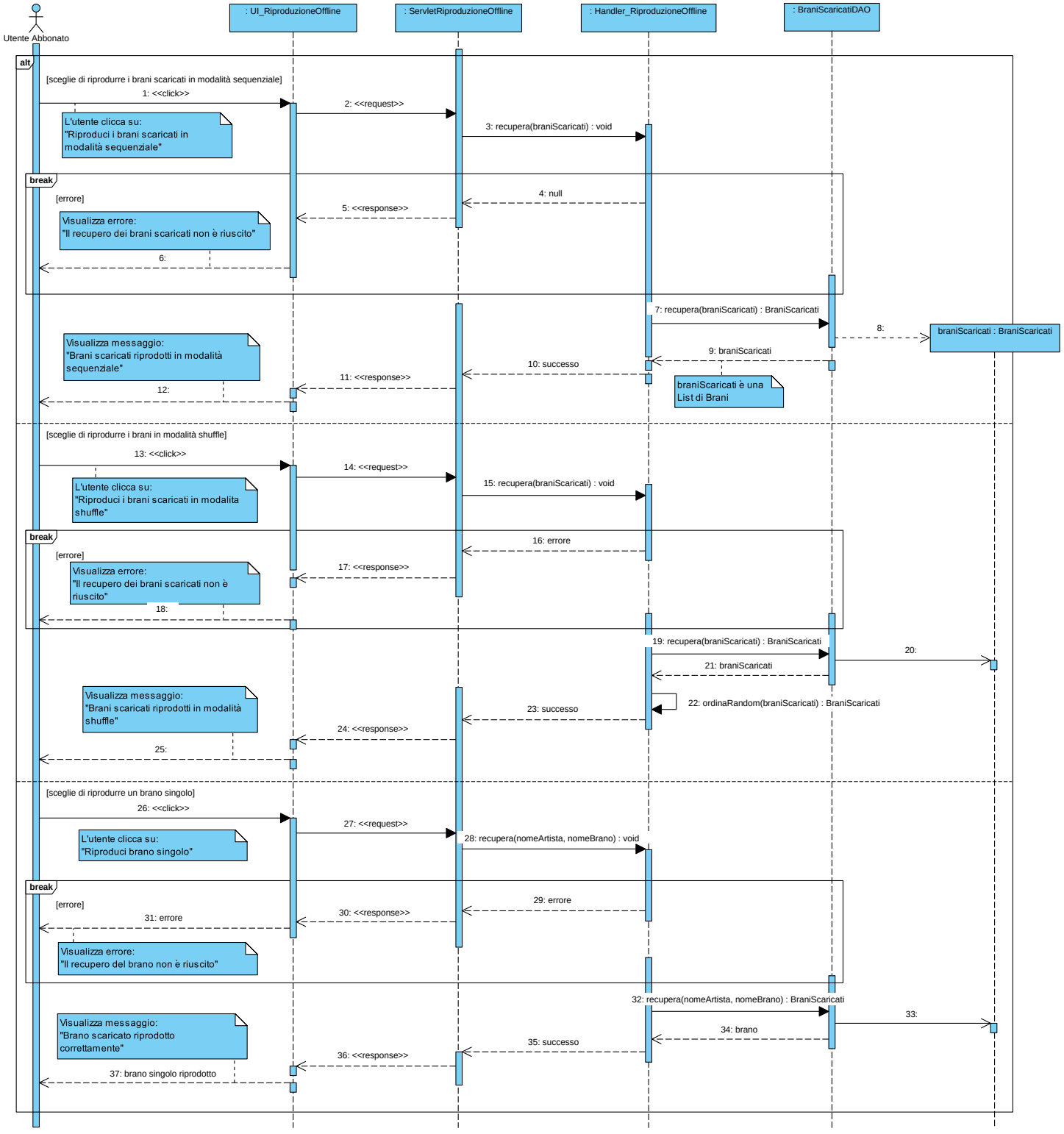


4.1.2 Sottosistema Gestione Riproduzione

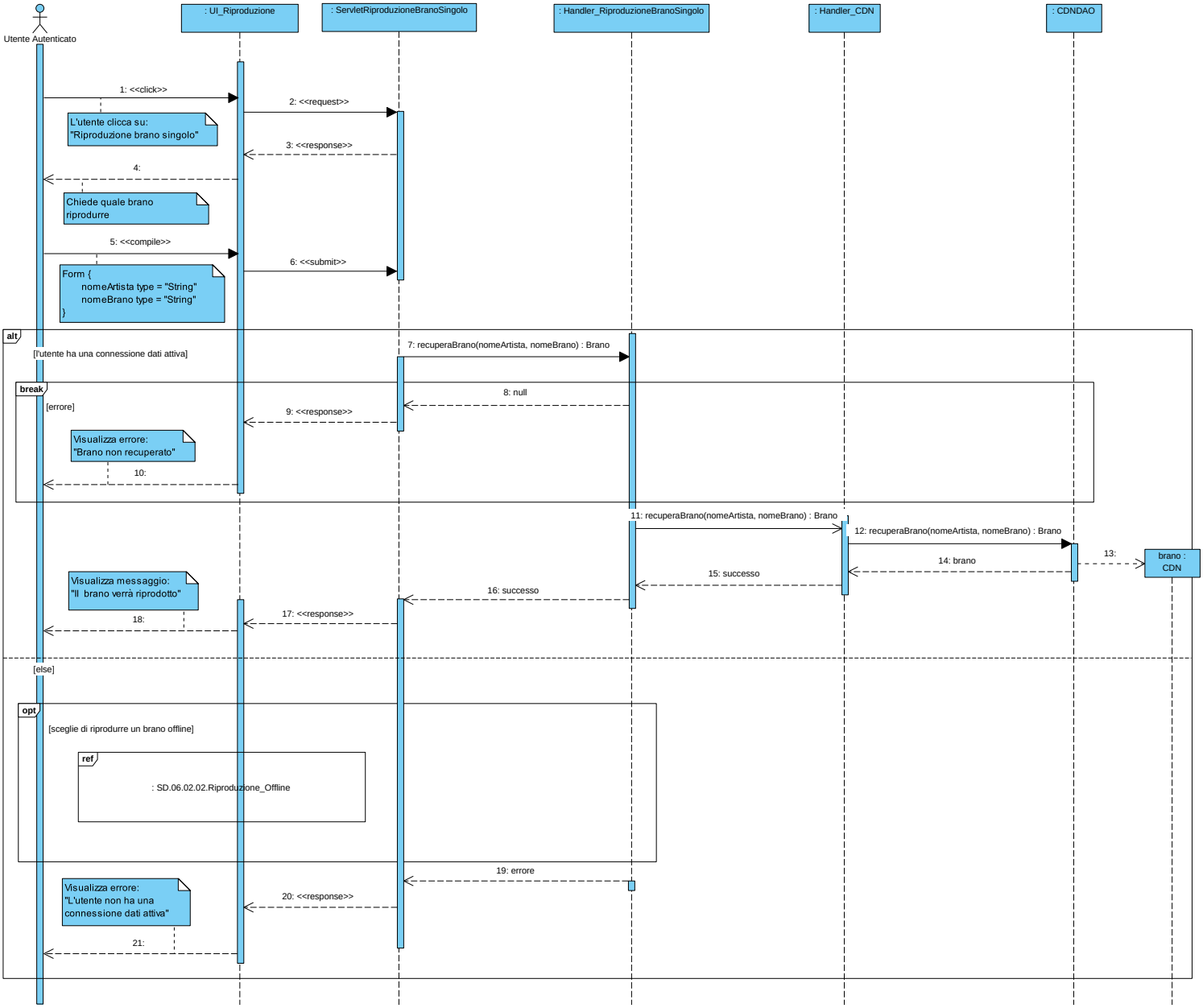
SDD.06.02.01.Effettua_Download



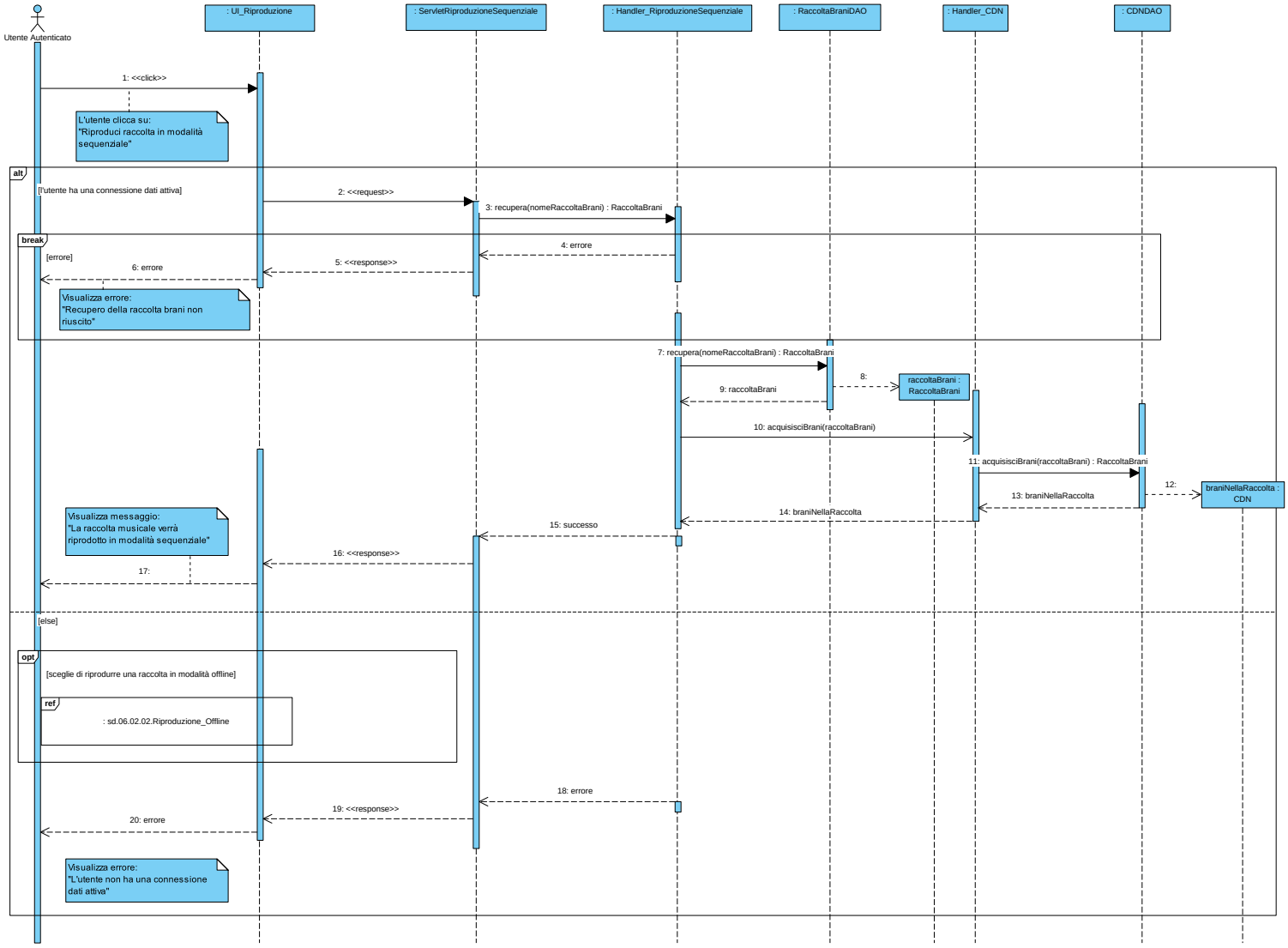
SDD.06.02.02.Riproduzione_Offline



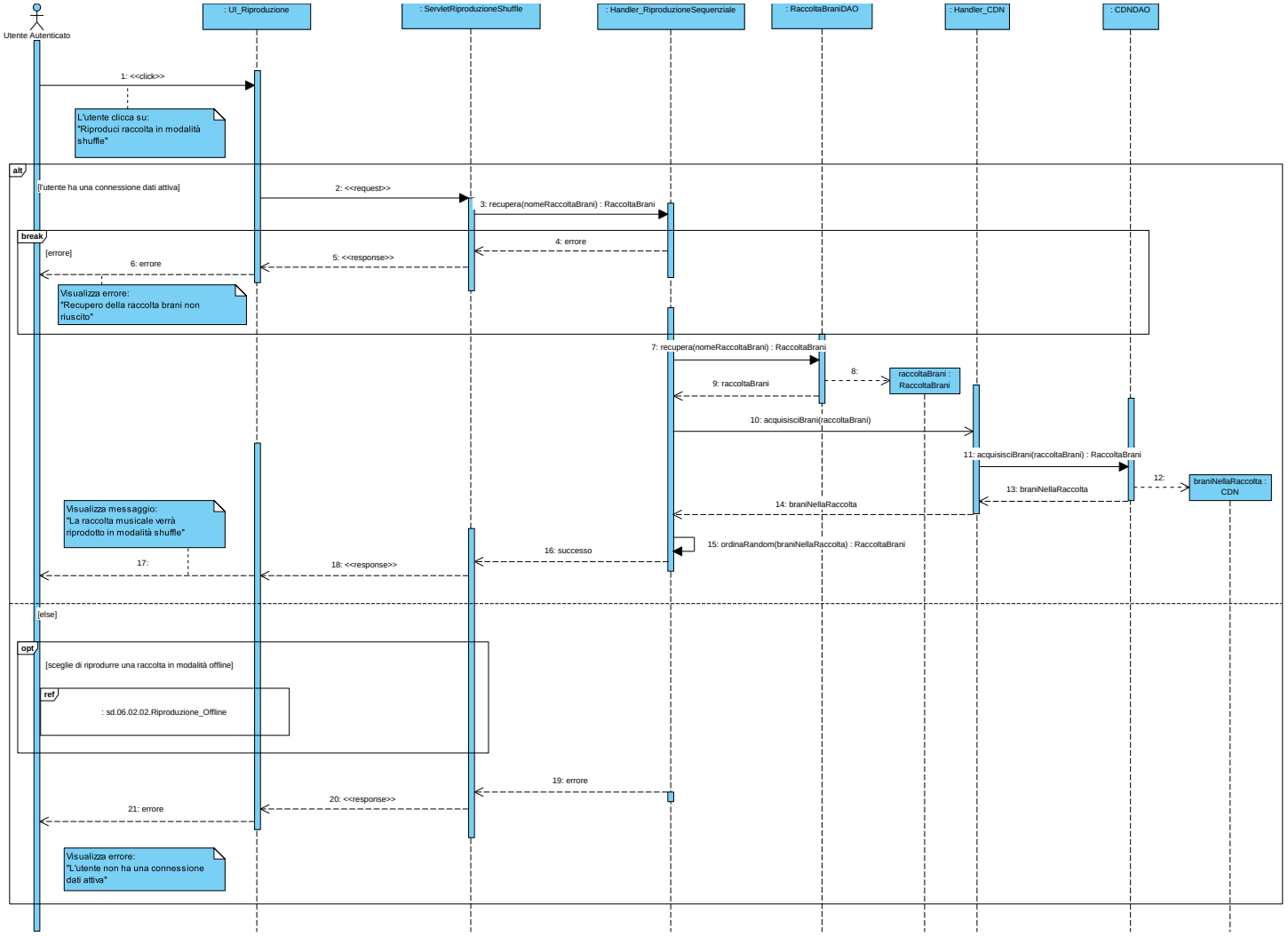
SDD.06.02.03.Riproduzione_BranoSingolo



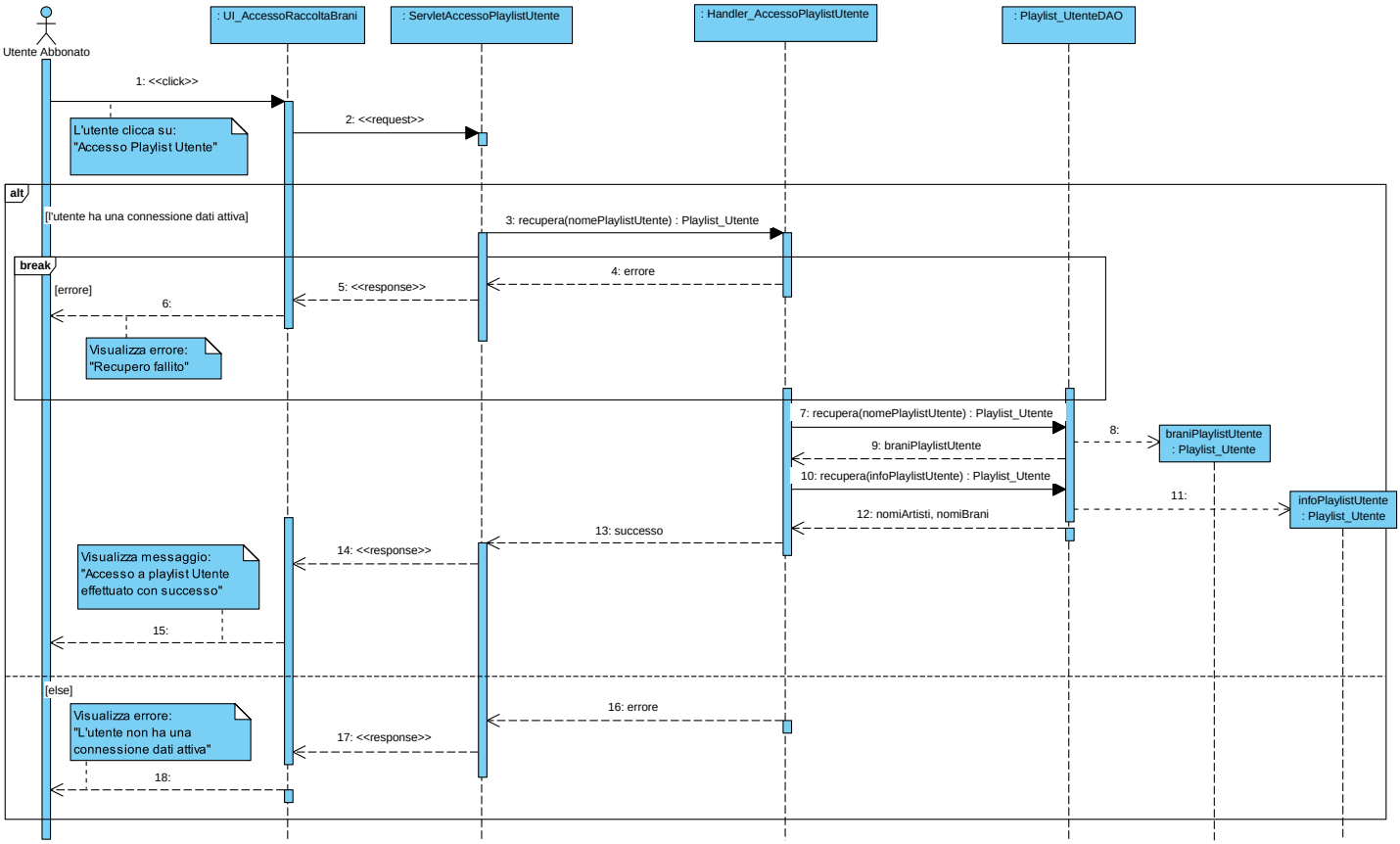
SDD.06.02.04.Riproduzione_Sequenziale



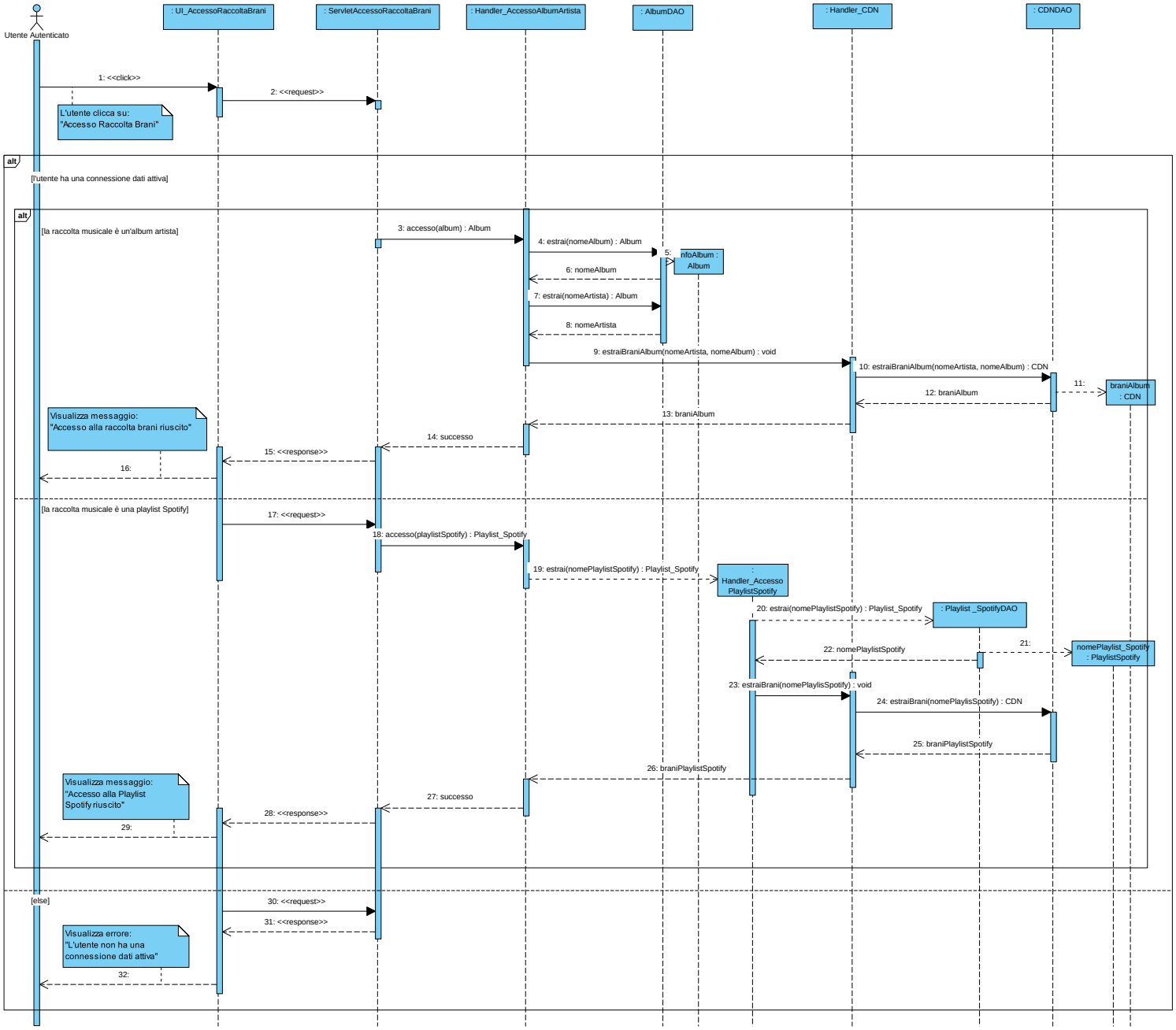
SDD.06.02.05.Riproduzione_Shuffle



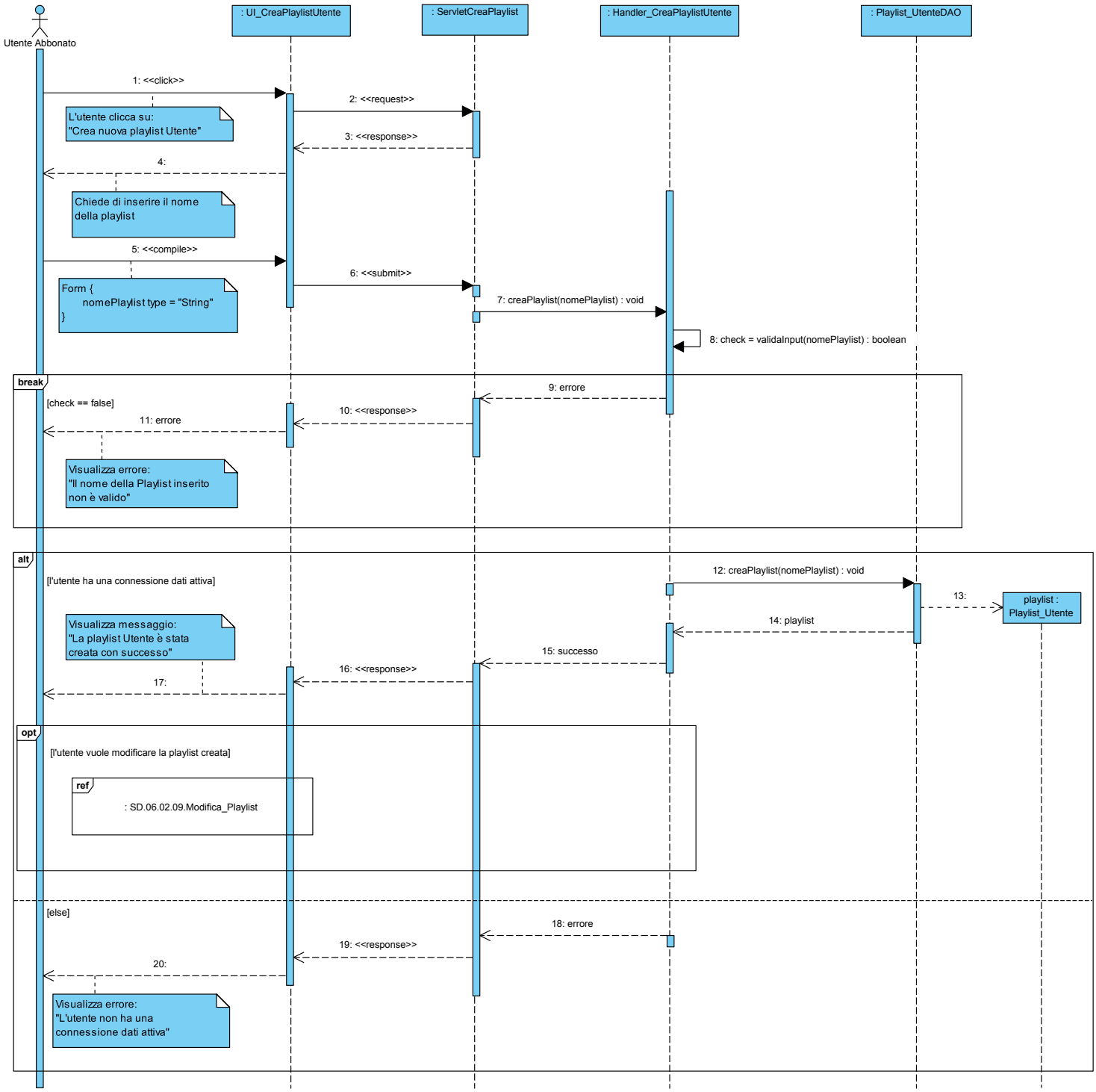
SDD.06.02.06.Accesso_PlaylistUtente



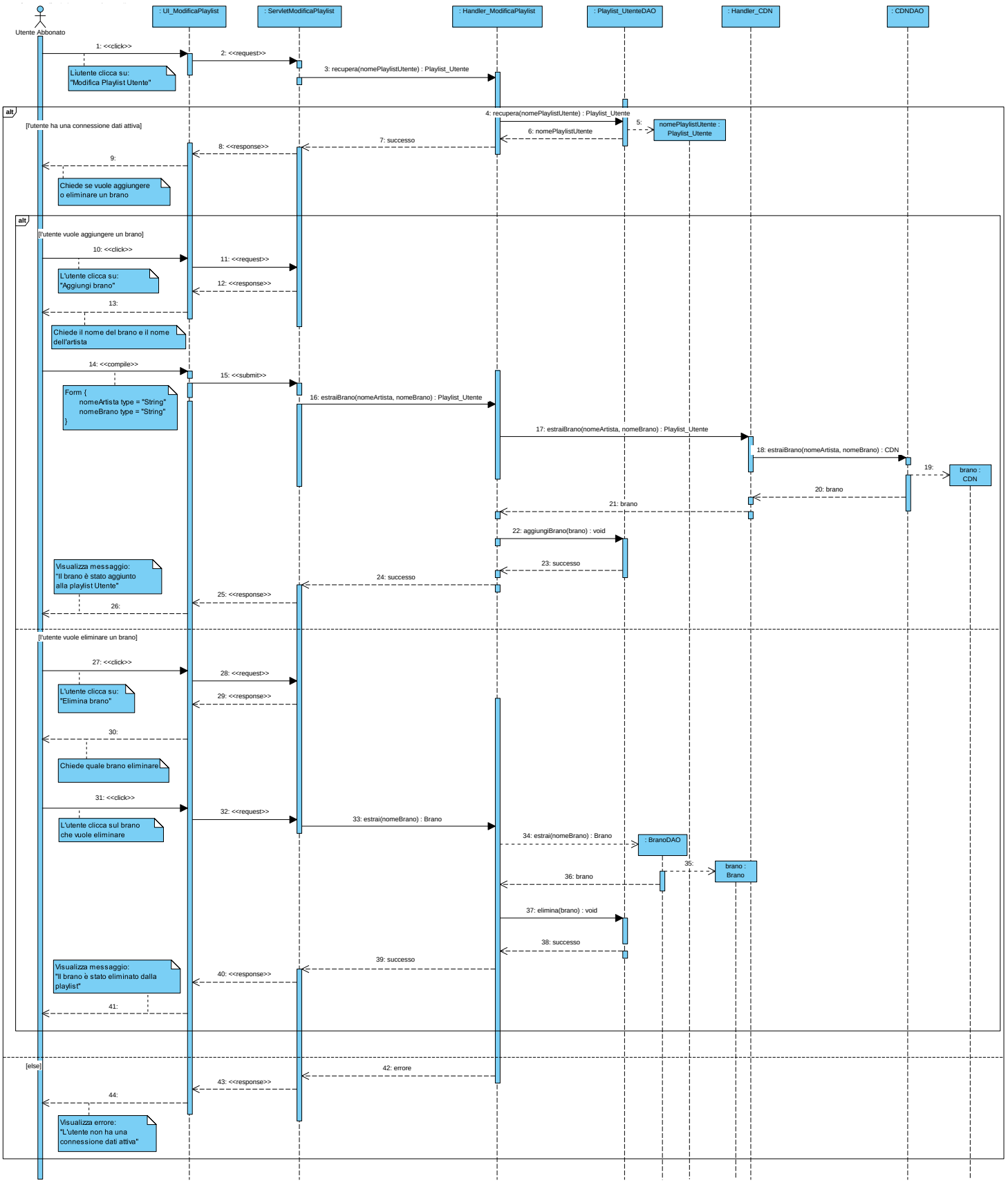
SDD.06.02.07.Gestione_AccessoRaccoltaBrani



SDD.06.02.08.Crea_Playlist

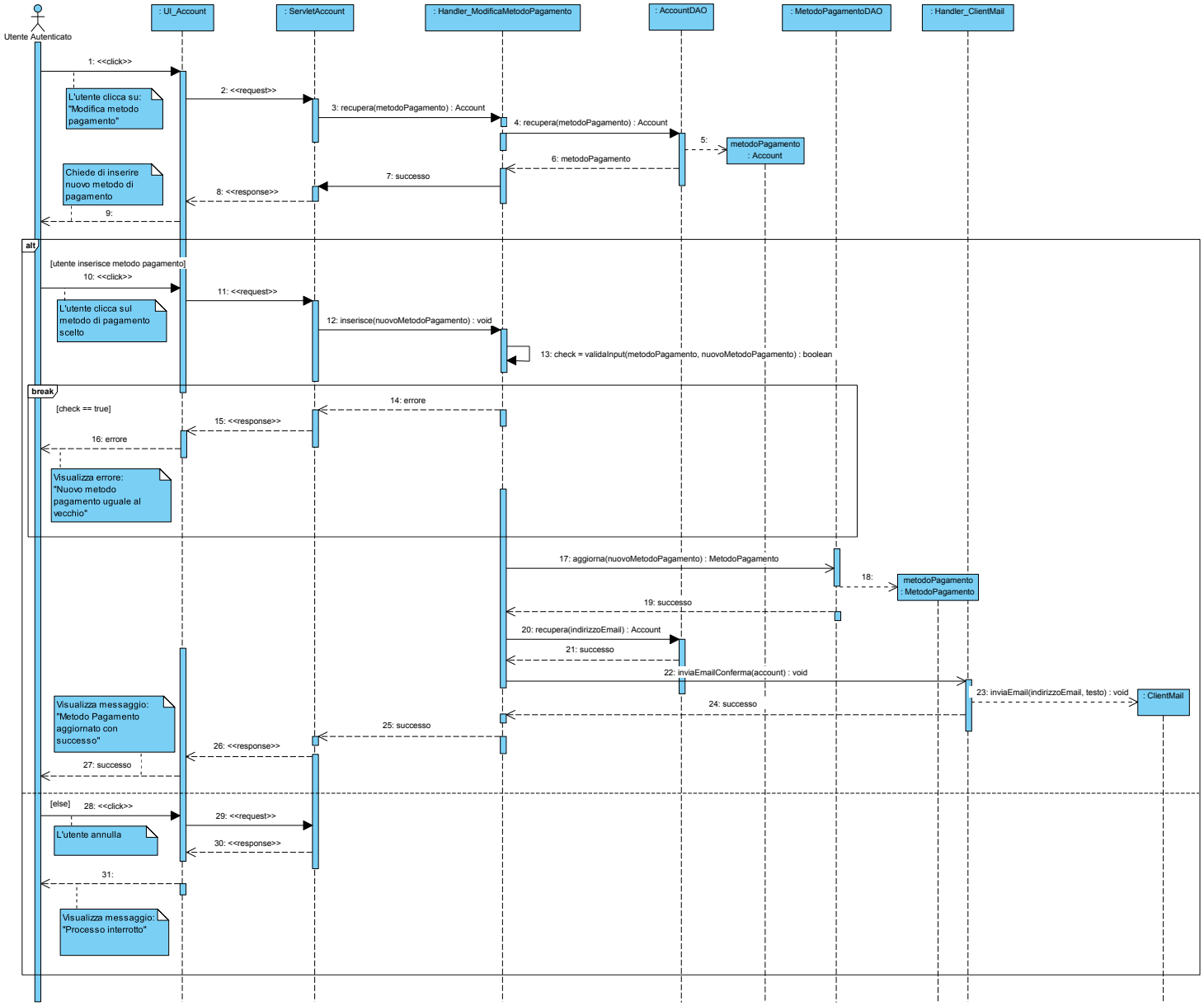


SDD.06.02.09.Modifica_Playlist

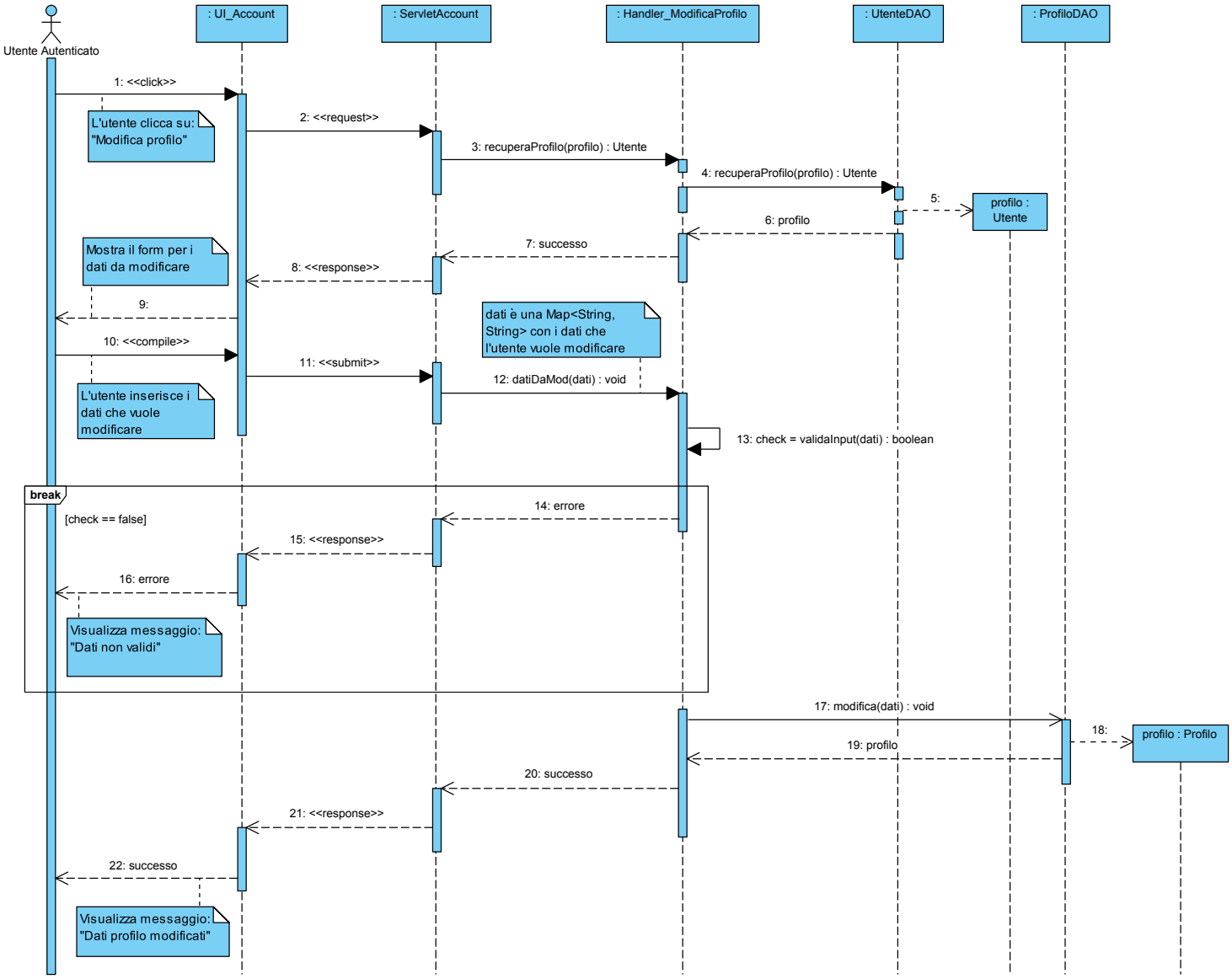


4.1.3 Sottosistema Gestione Account

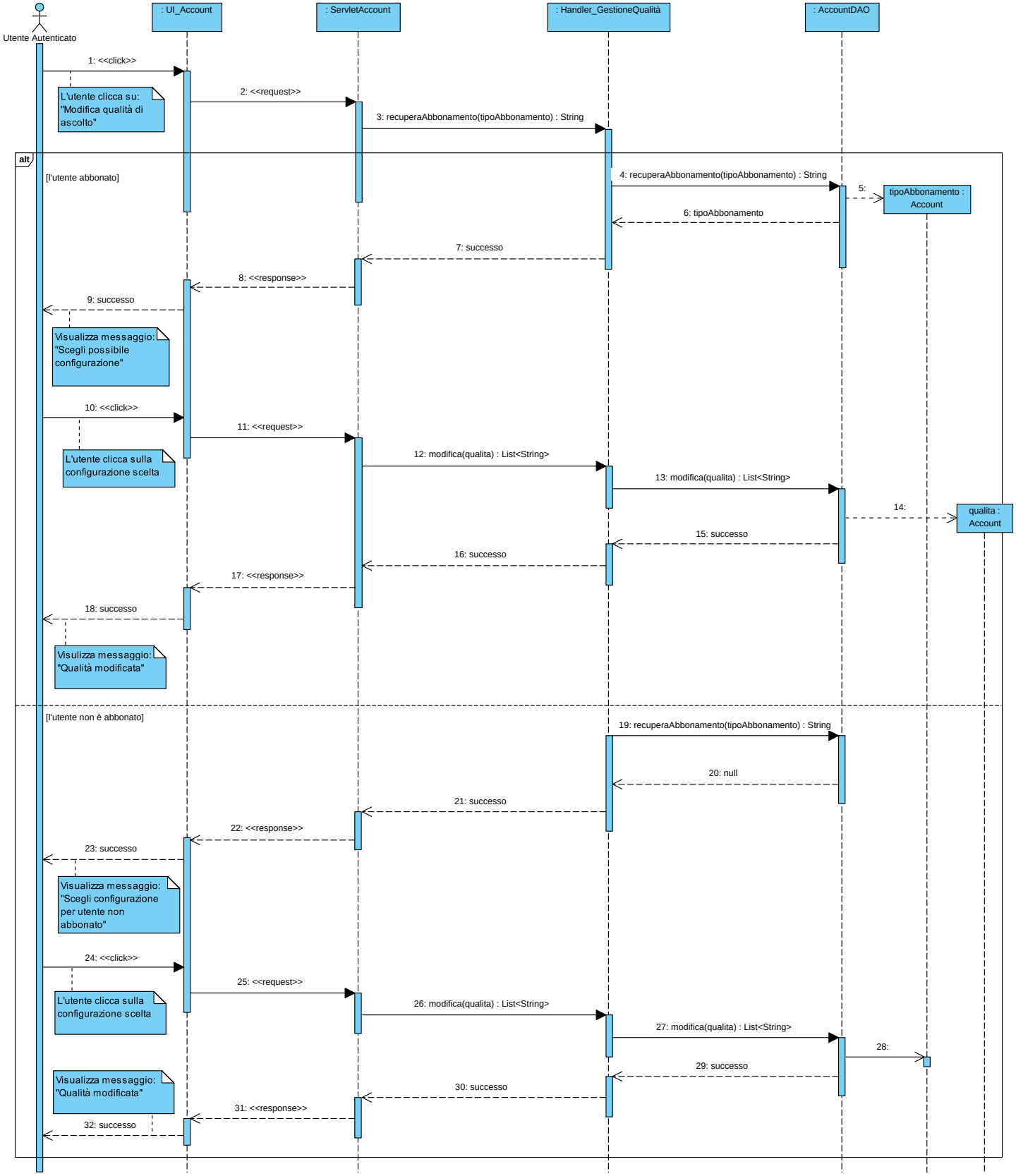
SDD.15.02.13.Modifica_MetodoPagamento



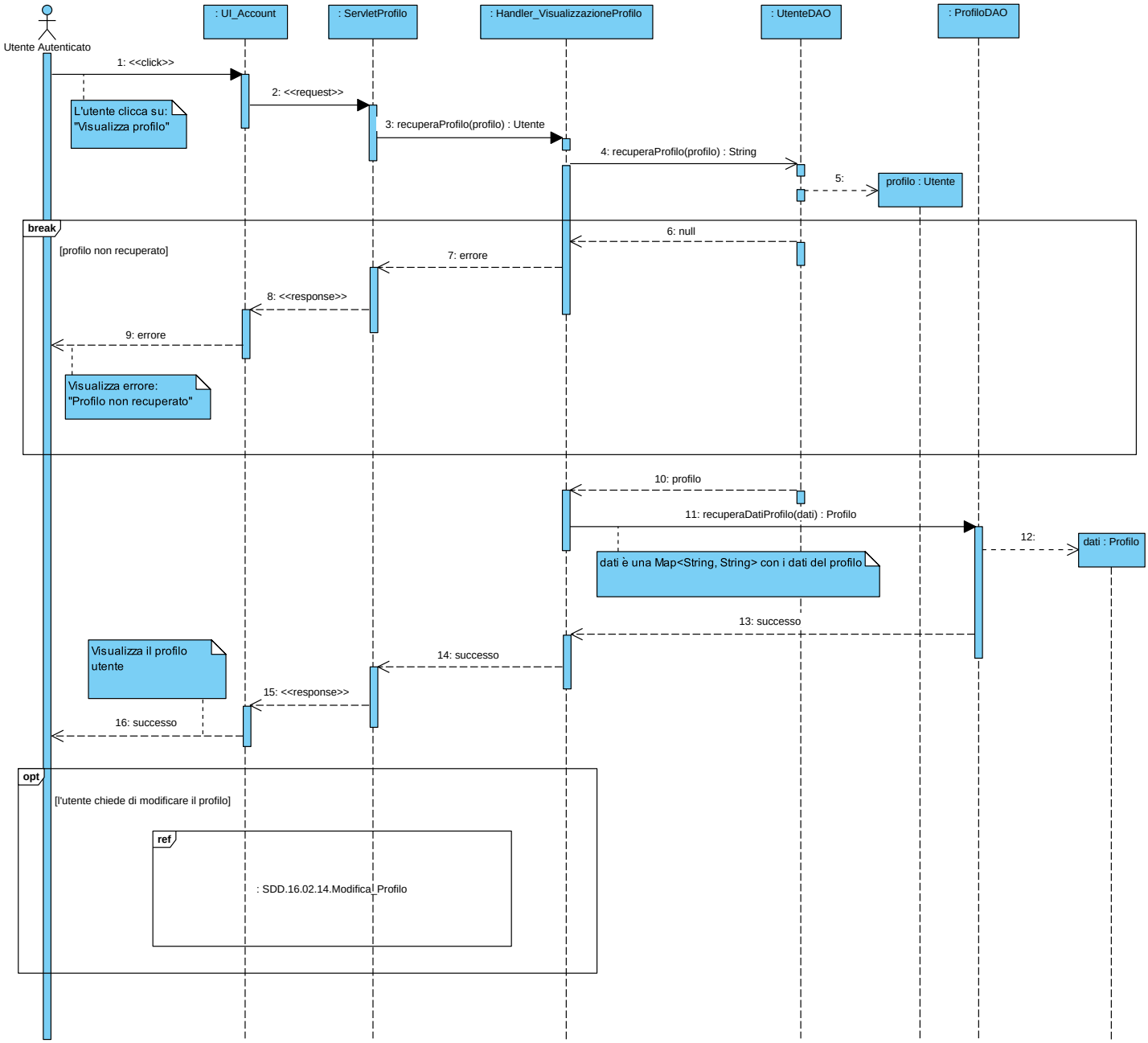
SDD.16.02.14.Modifica_Profilo



SDD.13.02.11.Modifica_GestioneQualitàRiproduzione

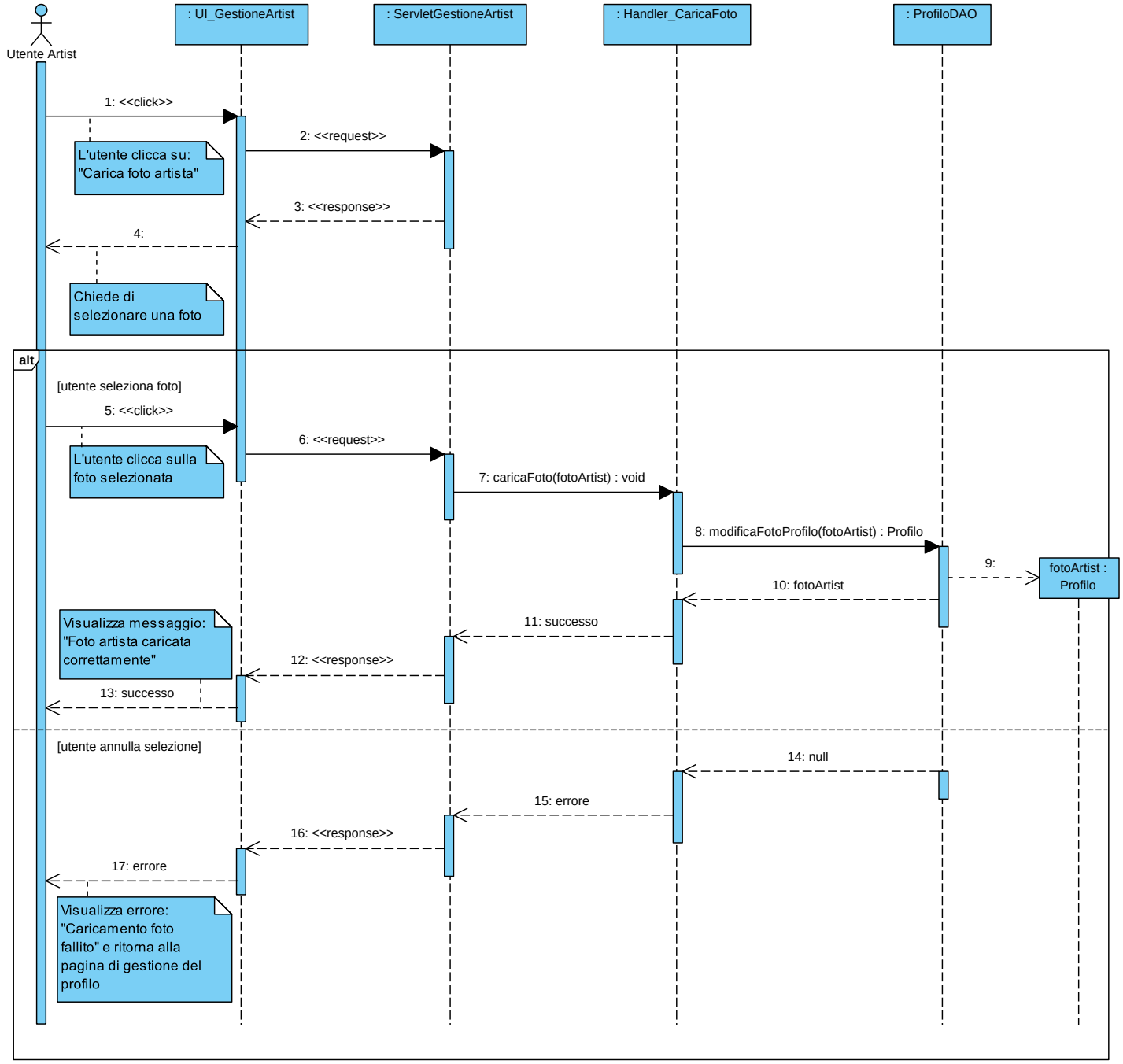


SDD.14.02.12.Visualizza_Profilo

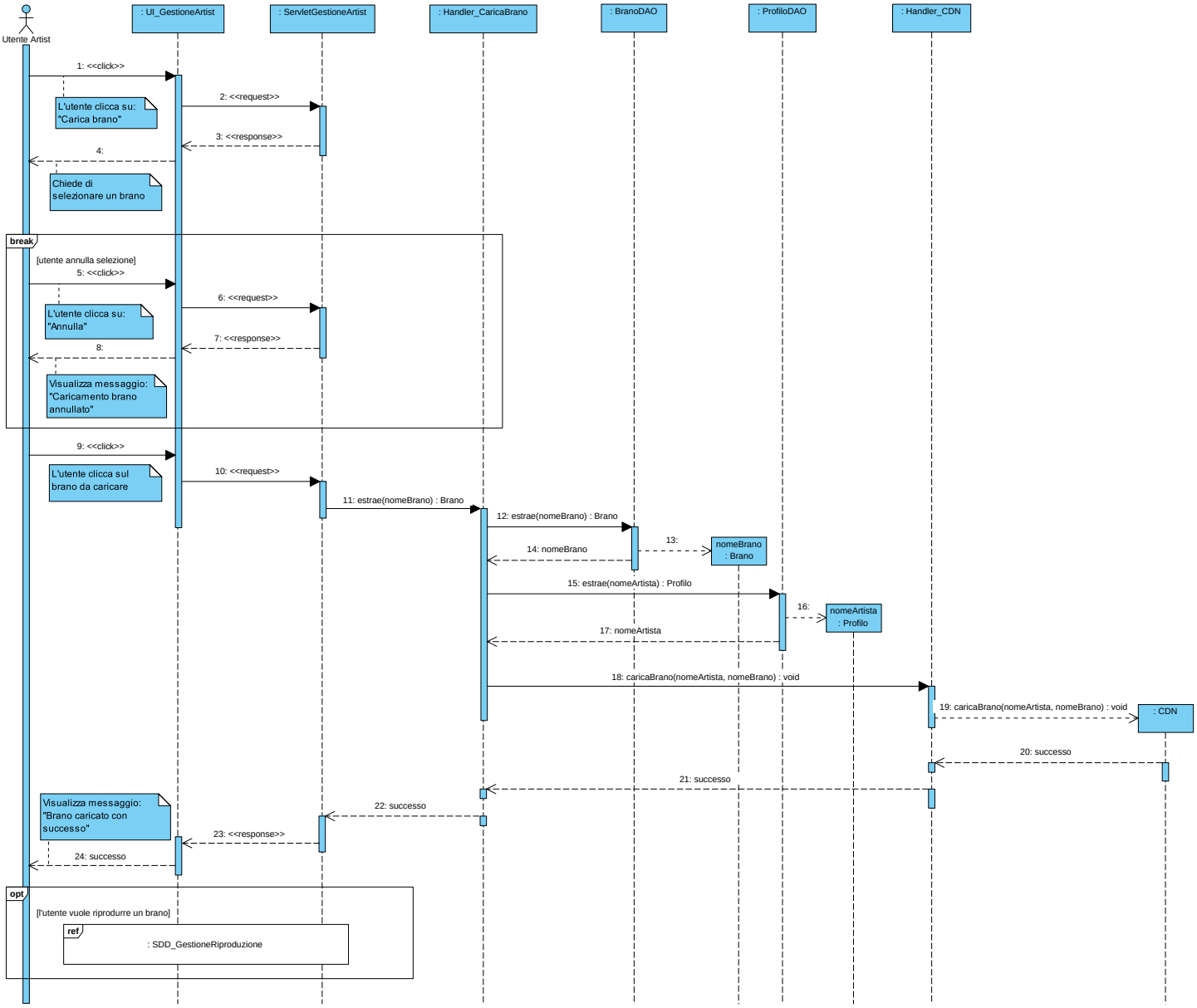


4.1.4 Sottosistema Gestione Artist

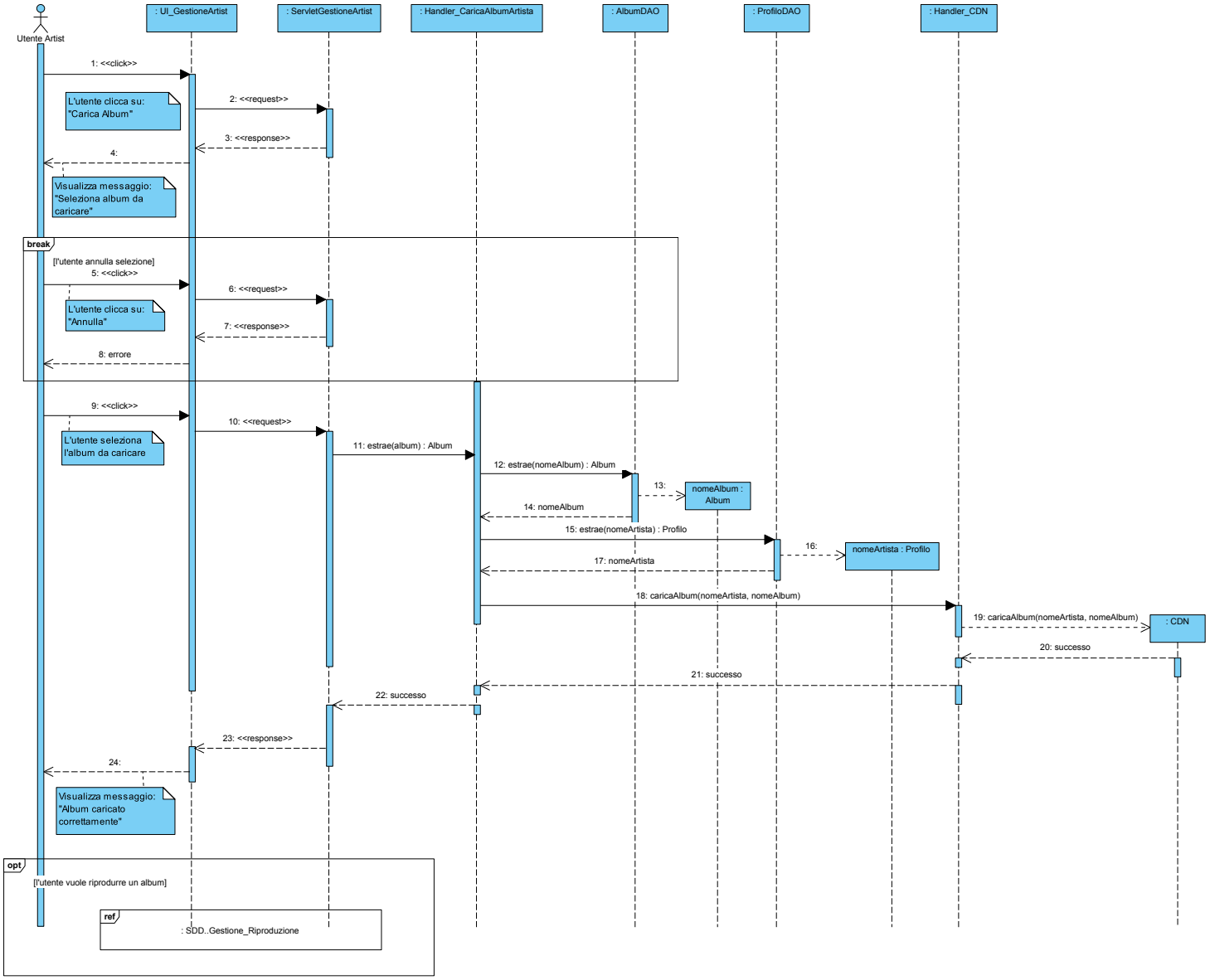
SDD.19.02.17.Carica_FotoArtista



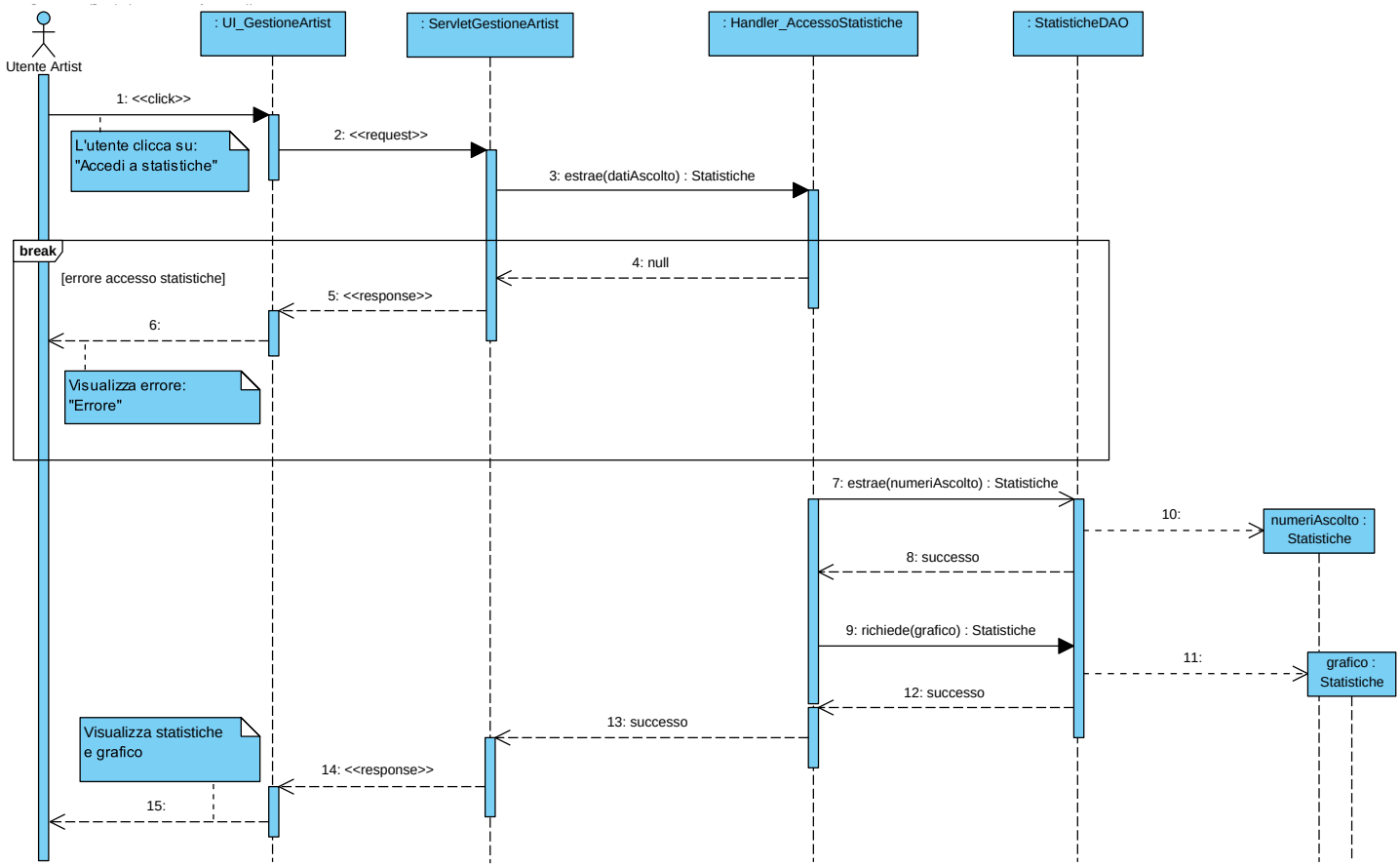
SDD.21.02.19.Carica_Brano



SDD.21.02.19.Carica_AlbumArtista

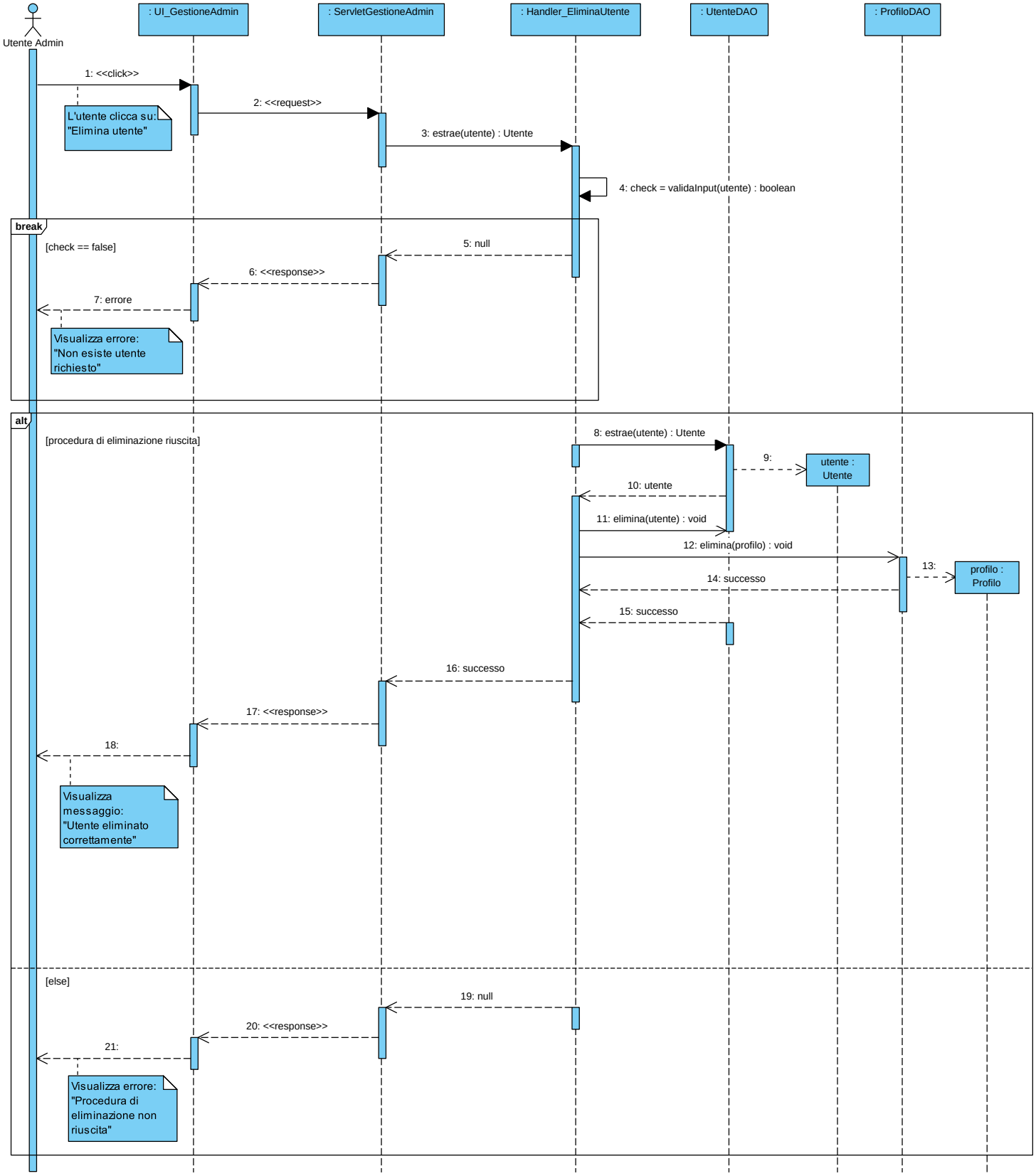


SDD.22.02.20.Accesso_Statistiche

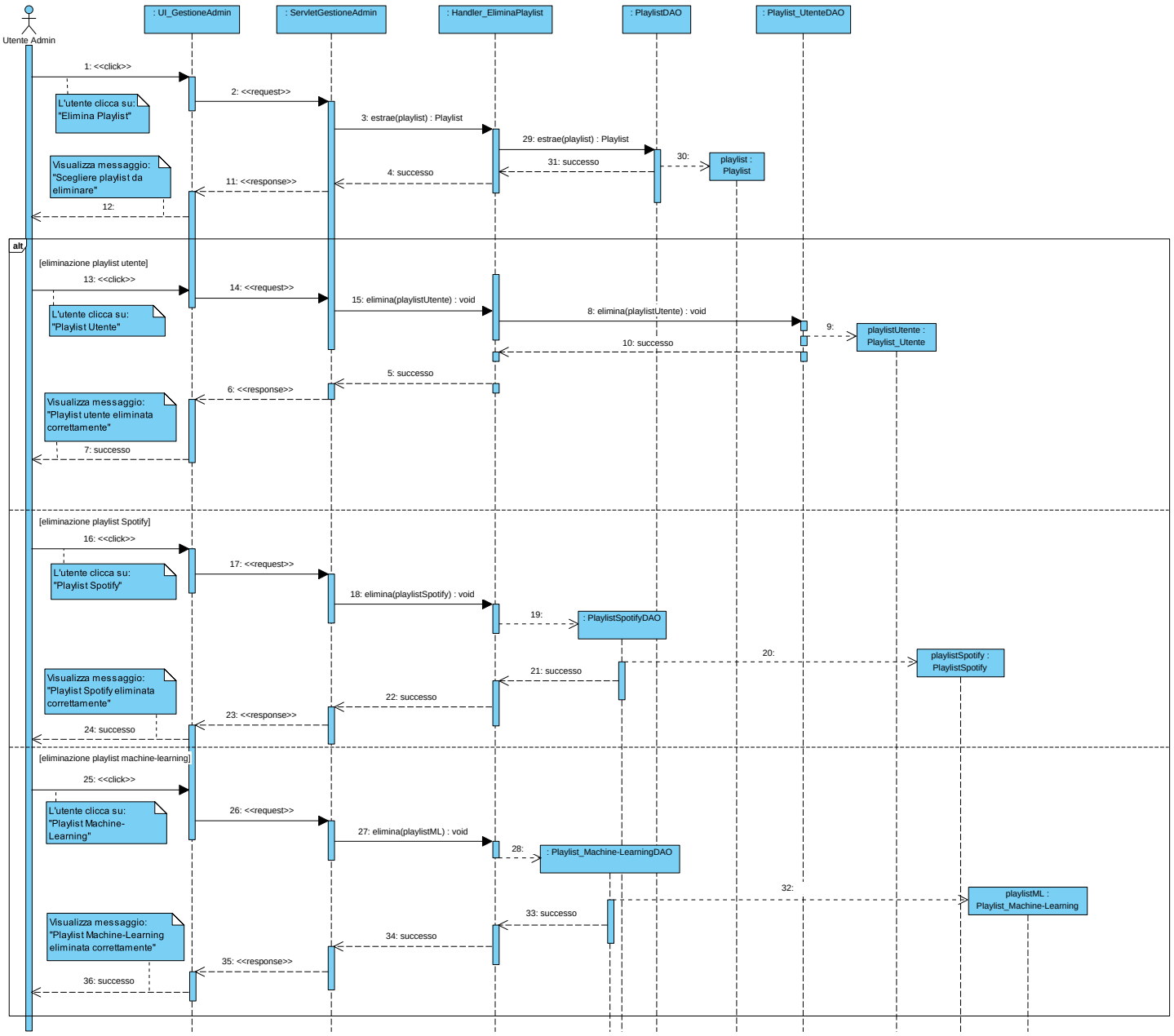


4.1.5 Sottosistema Gestione Admin

SDD.23.02.21.Elimina_Utente

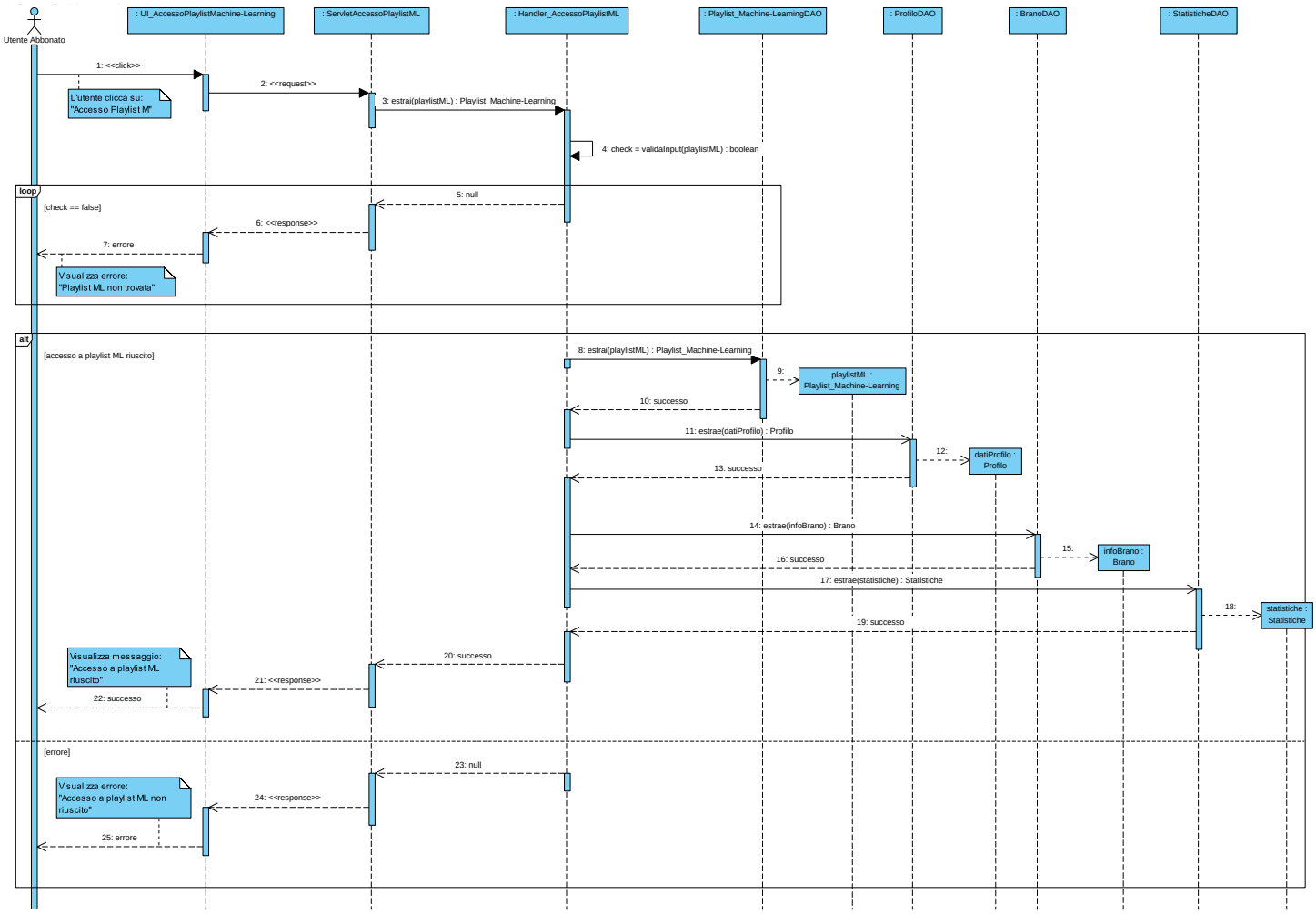


SDD.24.02.22.Elimina_Playlist

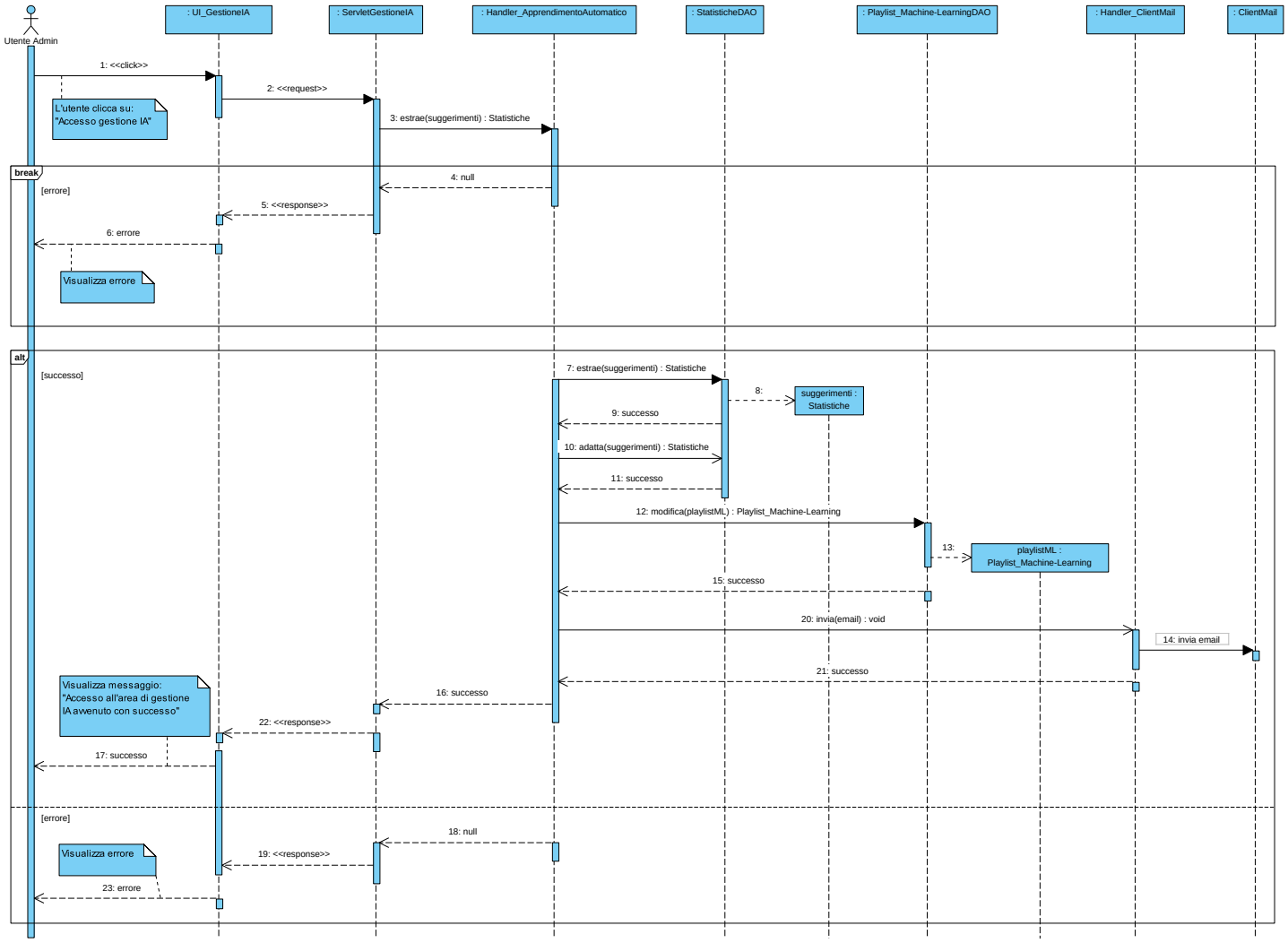


4.1.6 Sottosistema Gestione IA

SDD.25.03.01.Accesso_PlaylistML

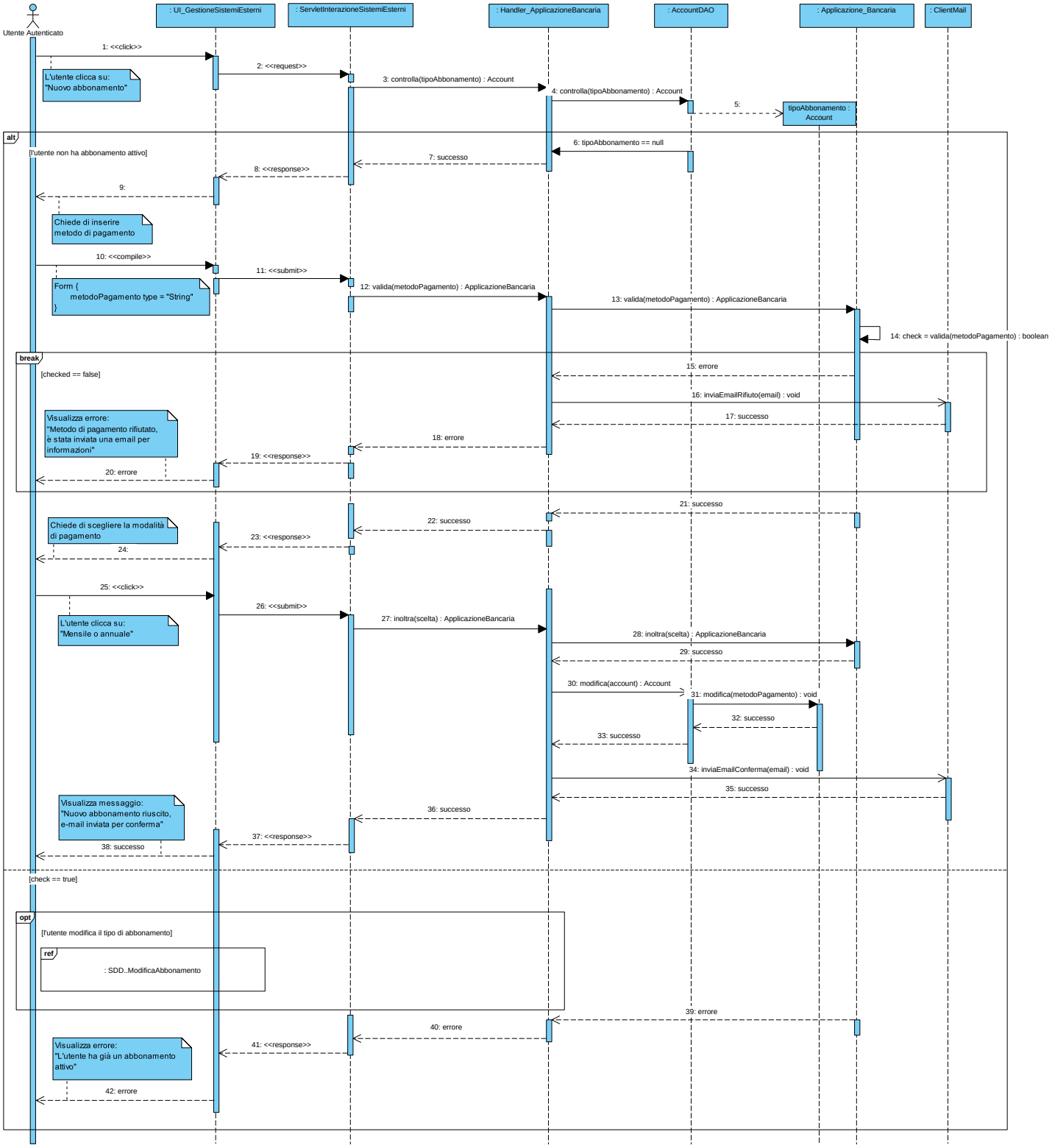


SDD.25.03.02.Apprendimento_Automatico

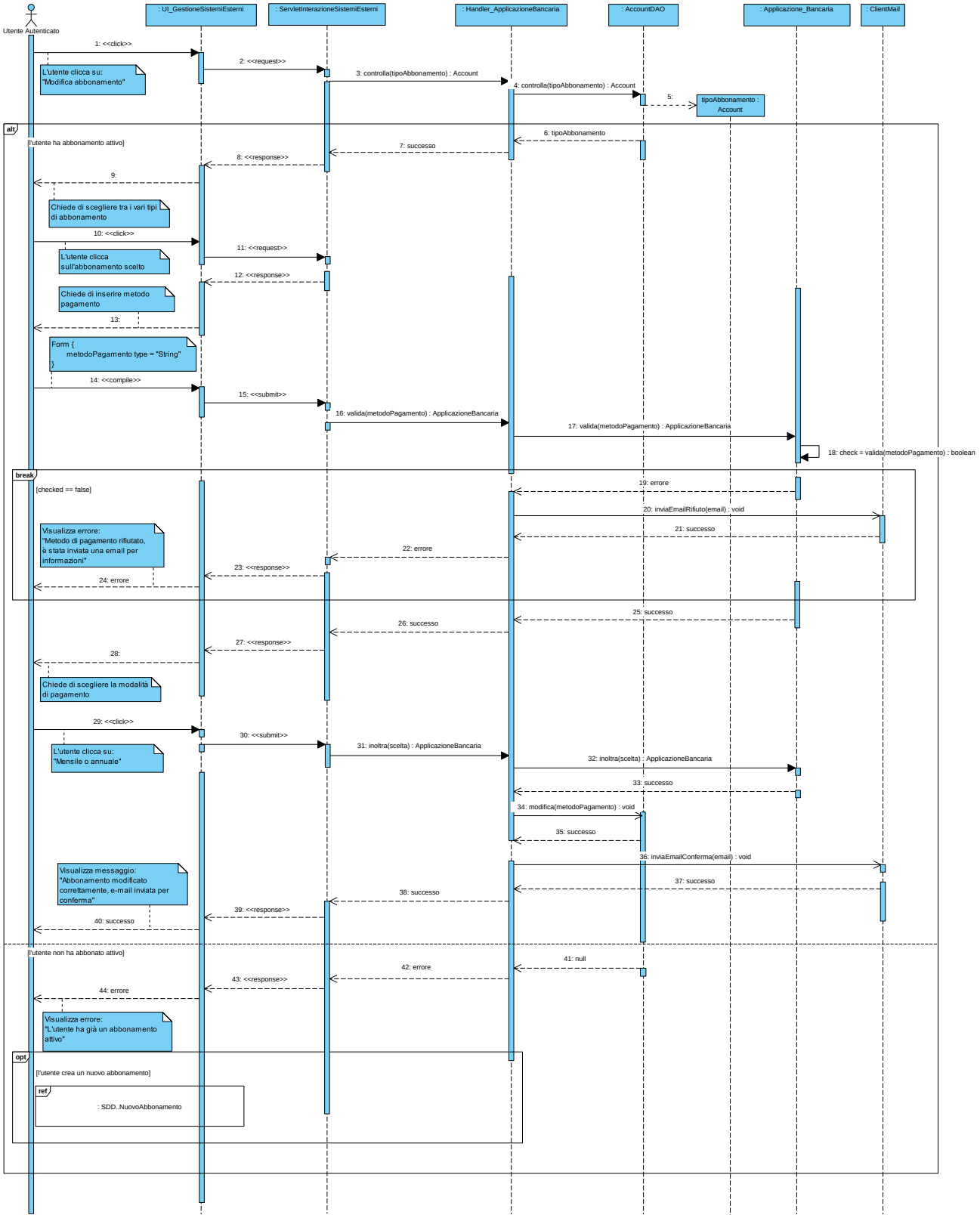


4.1.7 Sottosistema Interazione Sistemi Esterni

SDD.26.01.01.Nuovo_Abbonamento

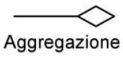


SDD.26.01.02.Modifica_Abbonamento



5. Diagramma delle classi Entity

Di seguito si illustra il diagramma delle classi Entity con le loro associazioni e molteplicità. Per facilitare la lettura del diagramma si illustrano e si spiegano in breve alcuni simboli utilizzati, per i dettagli si rimanda alla documentazione ufficiale del linguaggio.



: questo simbolo indica la relazione di *aggregazione* (chiamata anche *part-of*) e viene usata quando si lega un oggetto che rappresenta il *tutto* ad altri oggetti che rappresentano la *parte*.

: questo simbolo indica la relazione di *composizione* che è un tipo speciale di aggregazione in cui il *tutto* è un oggetto composto e le *parti* sono i suoi oggetti componenti.

: questo simbolo indica la relazione di *generalizzazione* (chiamata anche relazione *is-a*) e si usa quando ci si riferisce alla relazione tra due classi. La generalizzazione è quindi la relazione tra una classe e un'altra che ne è una versione più specifica.

5.1 Diagramma delle classi Entity in dettaglio