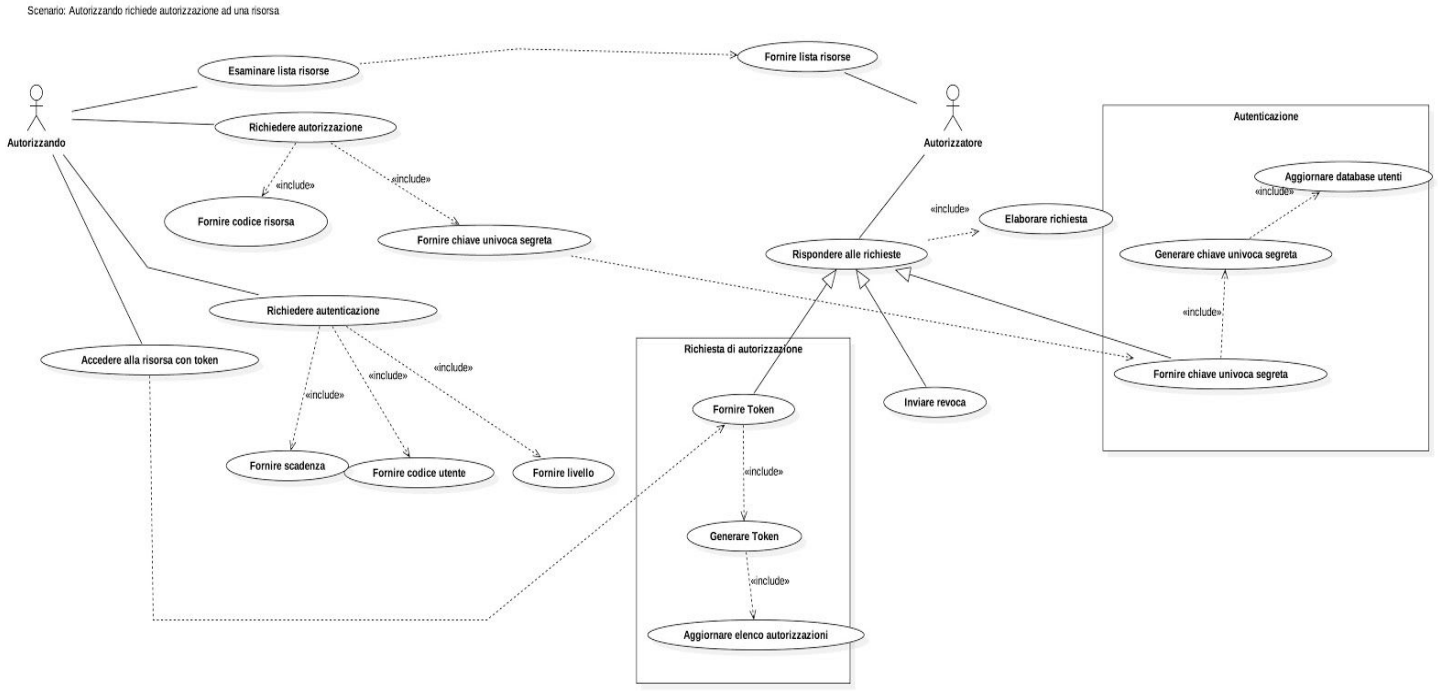
Anno accademico 2017-18
Prova finale di Ingegneria del software
Prof. Marco Brambilla, Como
#swEngComo

Gruppo 4
Tema A

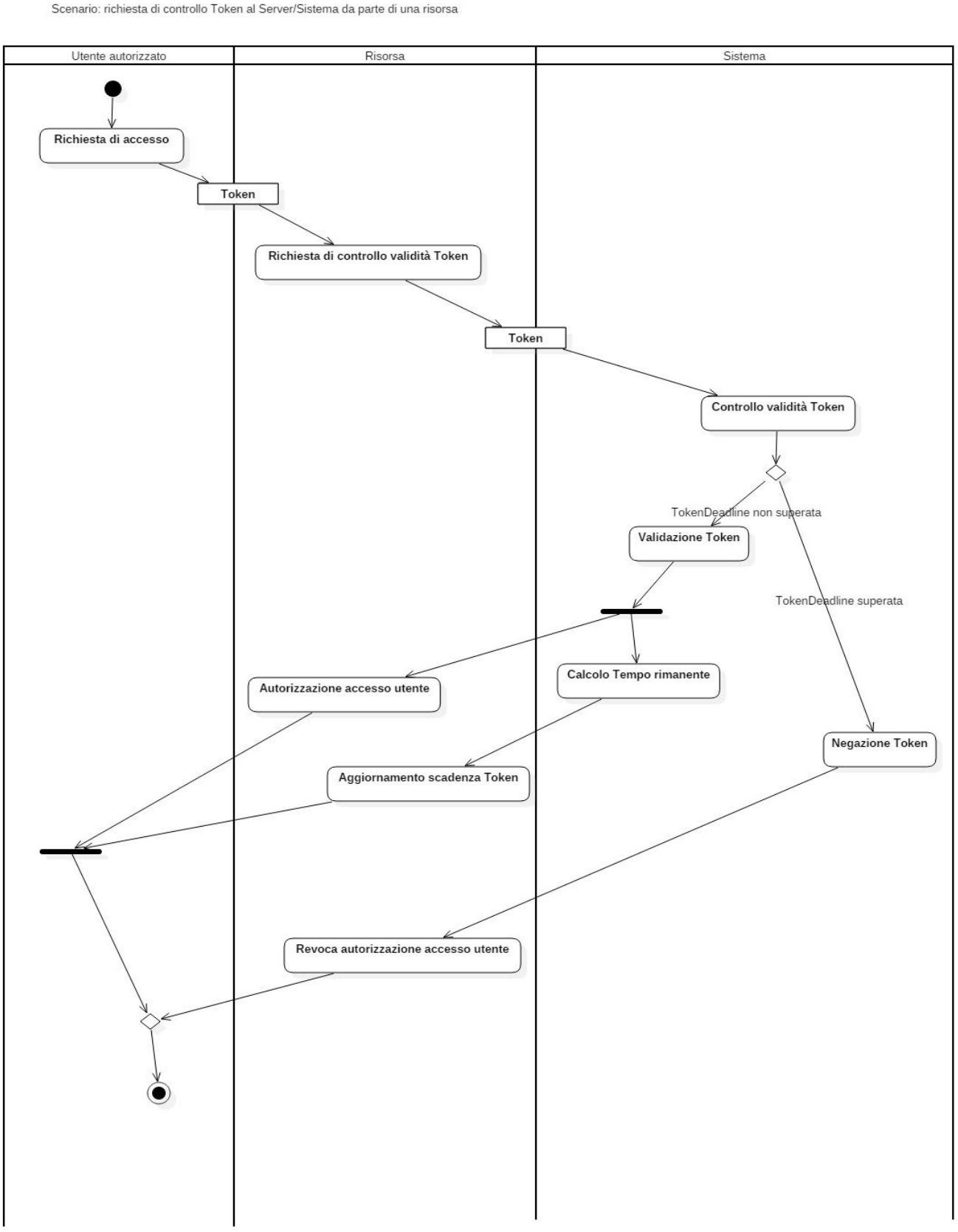
DESIGN: UML

Usecase diagram: scenario di autenticazione ed autorizzazione del Client



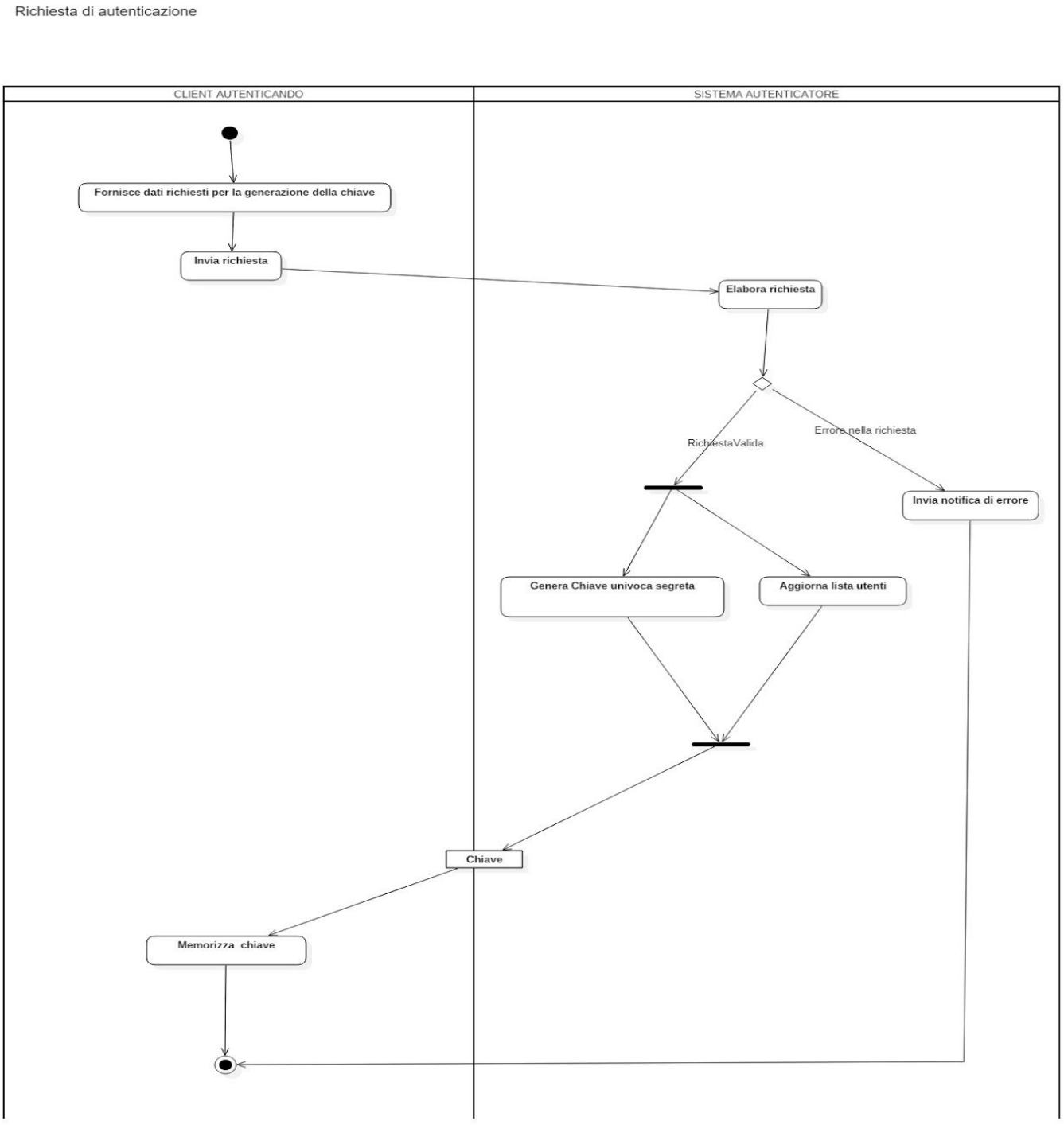
Si distinguono due sotto-gestioni all'interno dei casi d'uso del Sistema autorizzatore, distinti dall'elaborazione della richiesta: gestione della richiesta di autenticazione e quella di autorizzazione. Dalla prima dipenderà la richiesta di autorizzazione all'accesso di una risorsa da parte del Client, mentre dalla seconda dipenderà l'accesso diretto alla risorsa stessa.
N.B. L'accesso diretto alla risorsa con il Token non sarà implementata nel progetto, poiché fuori specifiche, ma rappresenta il comportamento tipico di un utente autorizzato in possesso del Token

Activity diagram: controllo validità Token da parte della Risorsa



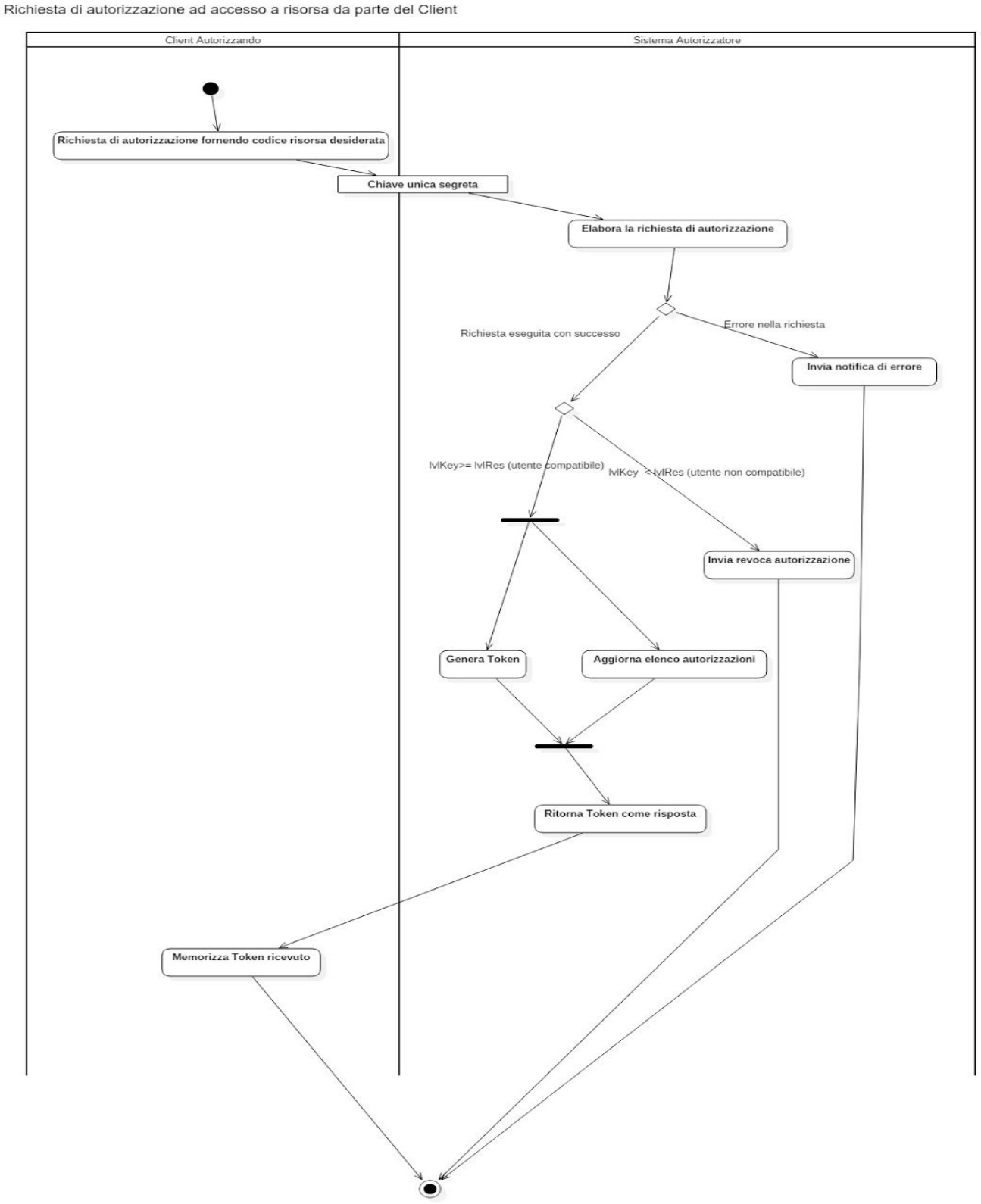
La risorsa può in qualsiasi momento controllare la validità del token ricevuto da un utente, passandolo al sistema che ne verifica la scadenza; il check del sistema ha 2 esiti: nel caso in cui il token risulti scaduto, viene inviata una notifica dal sistema alla risorsa associata al token, la quale provvederà ad informare l'utente della revoca , nell'altro caso, invece, il sistema fornisce una notifica di validità insieme al tempo rimanente di validità del Token.
N.B. le attività che svolge la risorsa, come da specifiche, non sono da implementare all'interno del nostro progetto. Il sistema deve essere pronto a gestire una richiesta di questo tipo.

Activity diagram: richiesta di autenticazione da parte del Client nei confronti del Sistema autenticatore



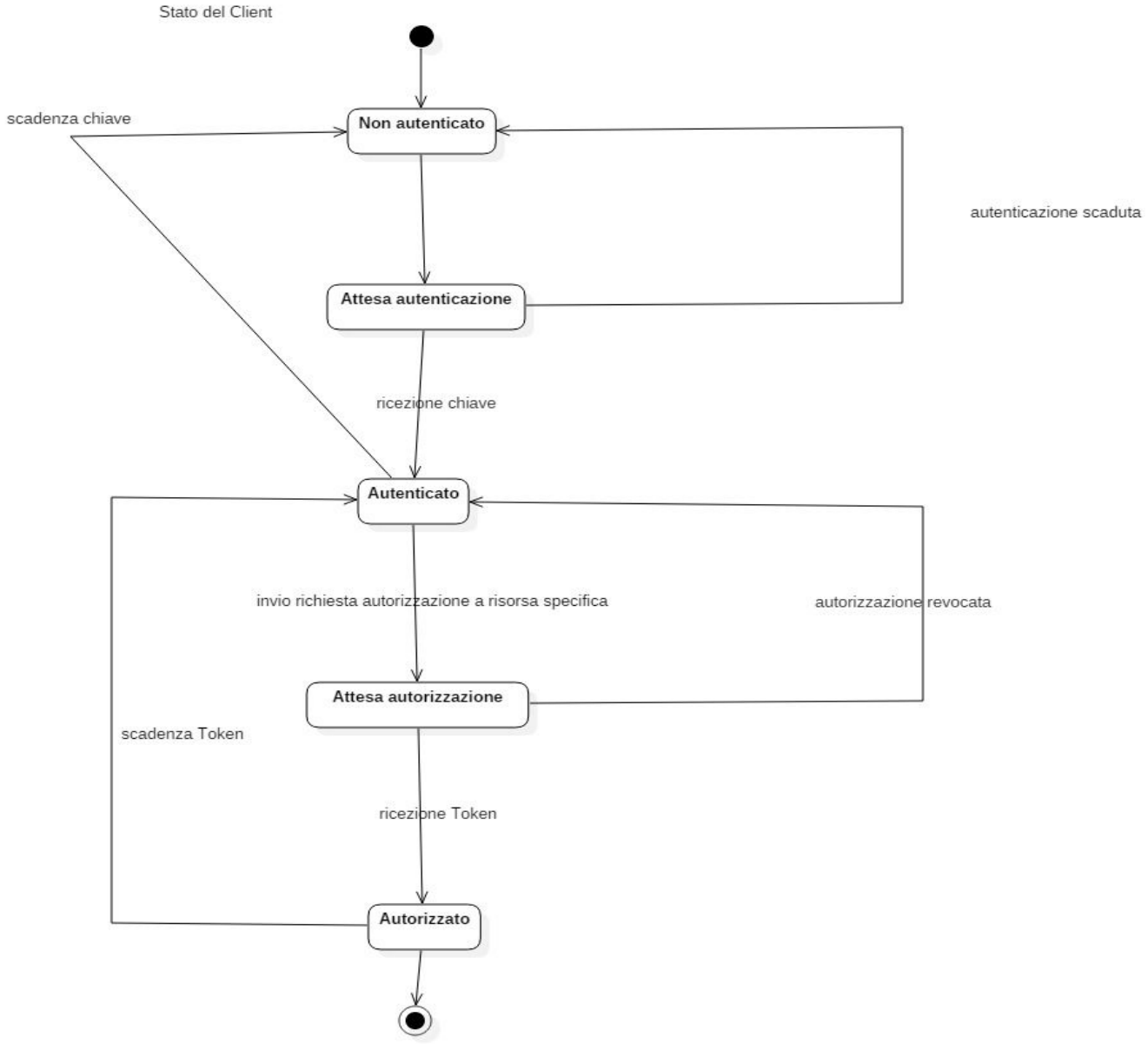
Indipendentemente dai dati forniti dal Client o da una sua precedente autenticazione il sistema, se la richiesta non genera errori di tipo ErrorType (vedi class diagram specifico tema A), il sistema genera sempre una nuova chiave univoca segreta e la associa ad un utente aggiornando (sovrascrivendo precedenti autenticazioni) la lista degli utenti autenticati.
La chiave viene memorizzata dal Client non appena ricevuta come risposta alla richiesta.

Activity diagram: richiesta di autorizzazione



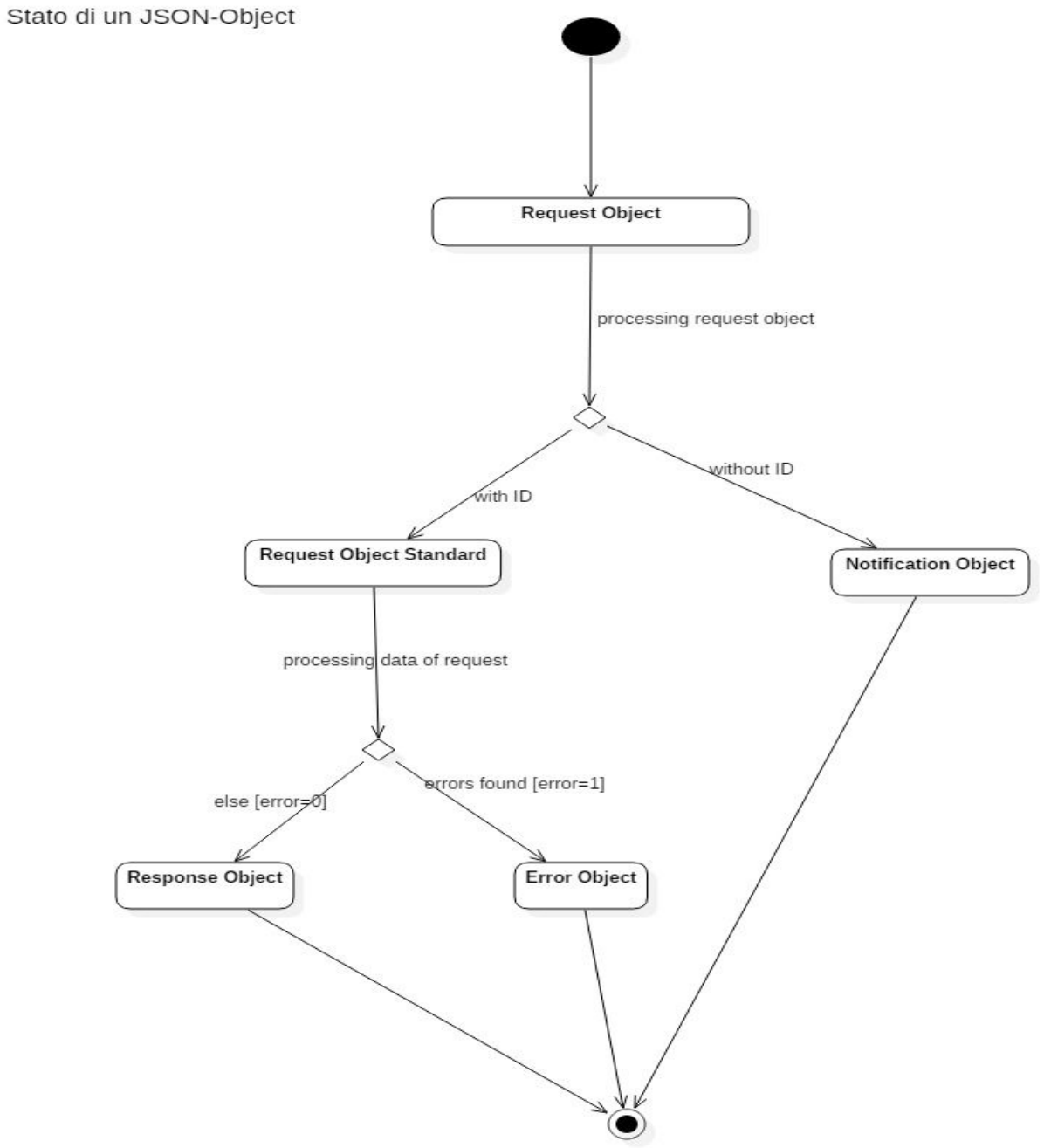
In base al confronto tra il livello della risorsa richiesta e della chiave passata dal Client, il sistema genera ed invia il Token o revoca l'autorizzazione all'accesso.

StateChart diagram: stato del Client autorizzando



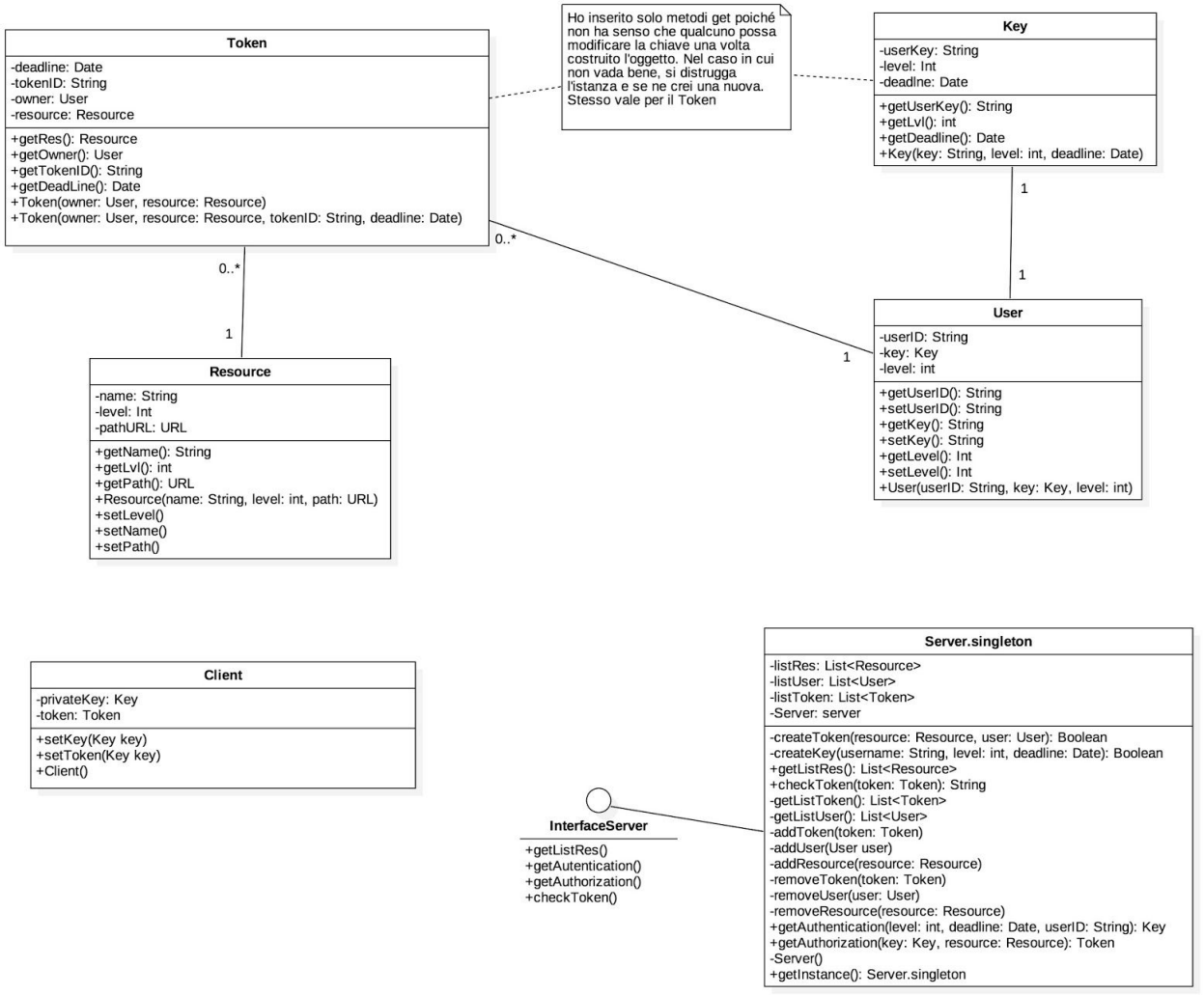
Il diagramma descrive le modifiche di stato dell'utente dalla condizione di non autenticato a quella di autorizzato, passando dalla fase di autenticazione e di autorizzazione.
N.B. La scadenza di un Token avviene sempre prima di quella della chiave: ovvero da Autorizzato a Non autenticato si passa sempre attraverso lo stato di Autenticato

Statechart diagram: stato di un oggetto messaggio secondo il protocollo JSON-RPC



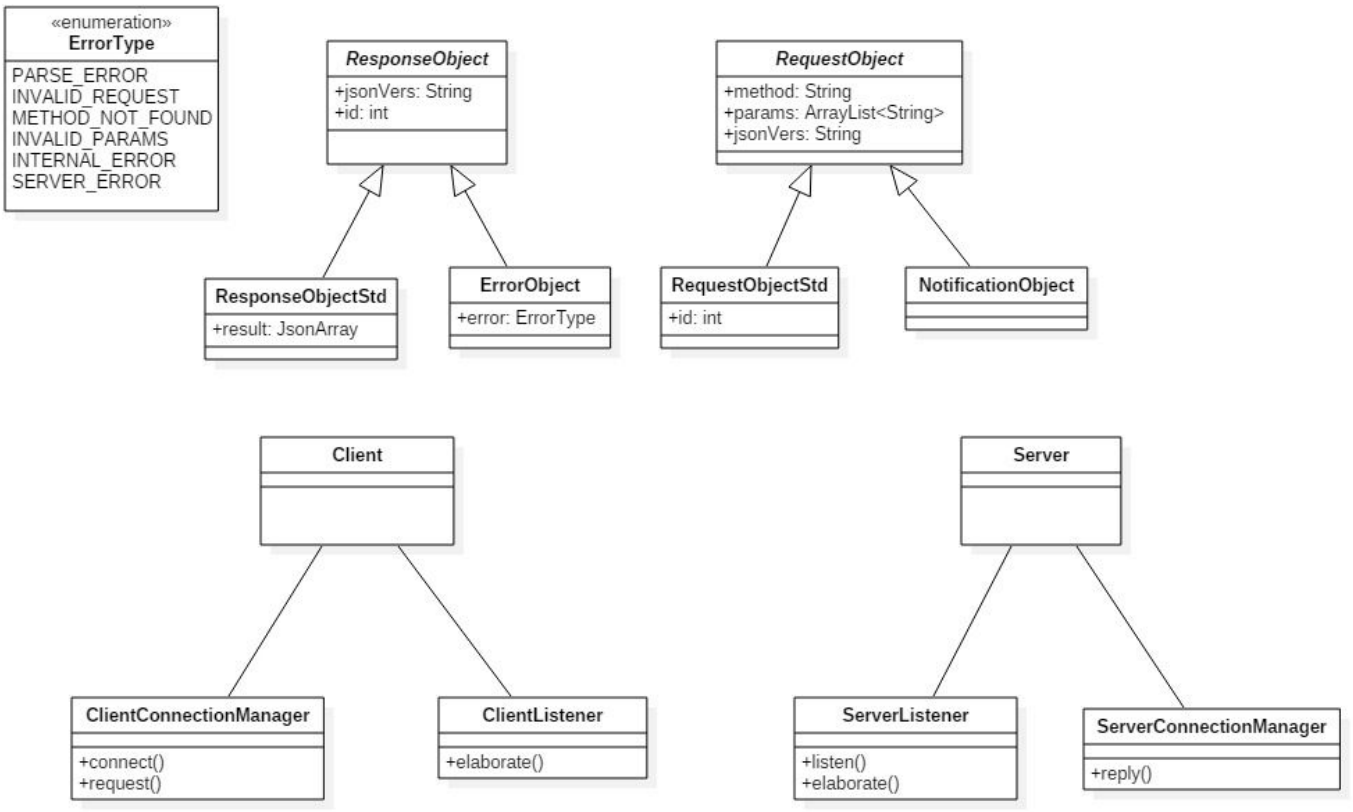
La presenza o meno dell'attributo id distingue la Notifica, la quale non necessita di risposta (l'assenza di id rende non "rintracciabile" il Client di partenza), dalla Richiesta standard che invece necessita di una Risposta che si distingue in ErrorObject o ResponseObject, a seconda se il ricevente ha trovato errori (di diverso tipo: vedi ErrorType in class diagram) o meno.

Class diagram: specifico Tema A



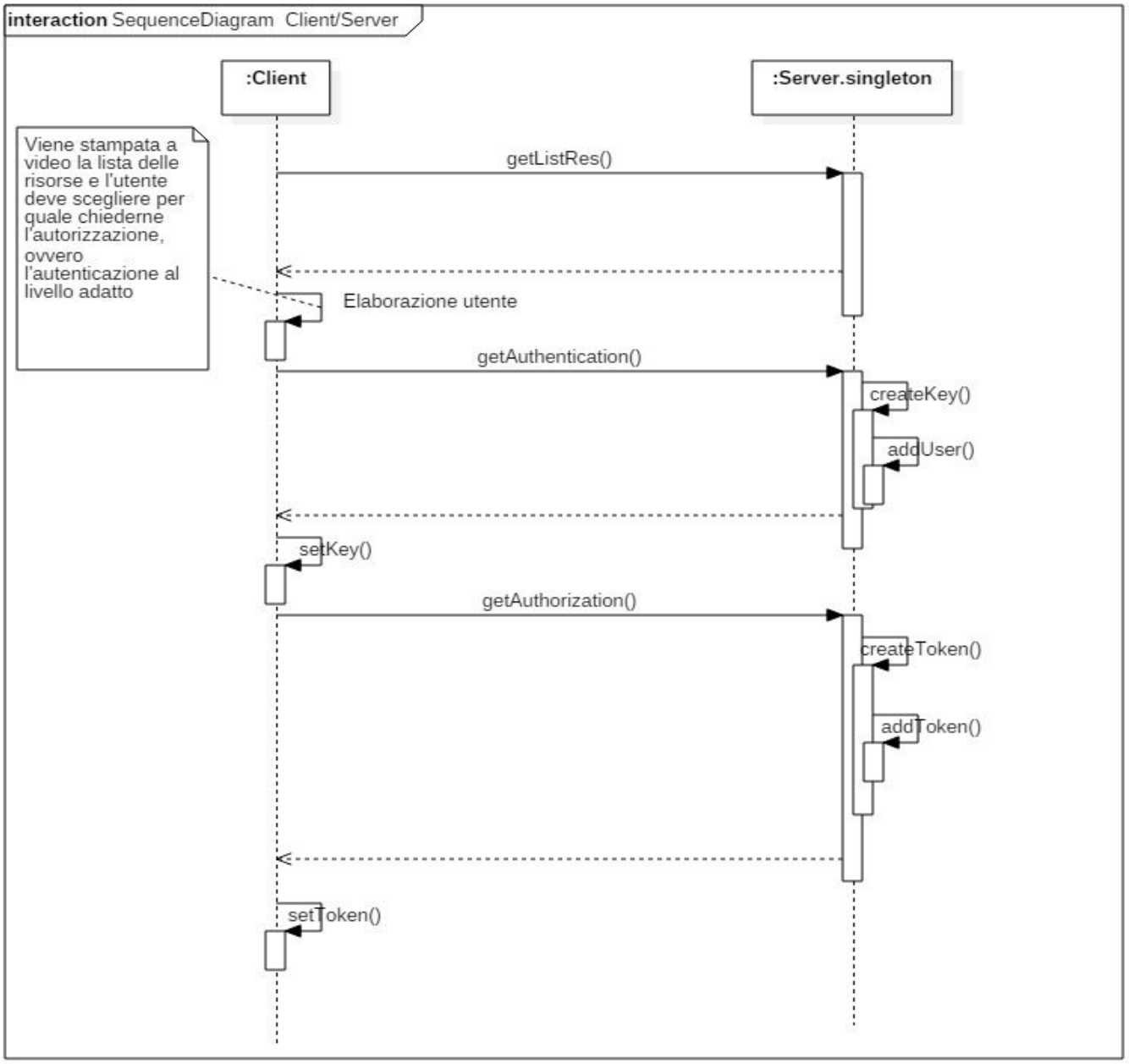
Il class diagram pensato per il tema è composto da due classi principali (*client* e *server*) e da altre tipi di classi pensate come “entità”. Le classi *Token*, *User*, *Resource* e *Key* sono classi utilizzate per la gestione dei dati da parte delle istanze *Client* e *Server*, che avranno la responsabilità di organizzare ed elaborare tali dati.
Si è pensato di usare il Singleton come design pattern, in modo tale da garantire una sola generazione dell’istanza della classe *Server*.
Inoltre abbiamo pensato di far esporre alla classe *Server* una interfaccia con tutti i metodi utilizzabili dall’esterno, in modo da rendere indipendente il codice e le altre classi dalla classe *Server.singleton*.
Le associazioni sono state pensate nel seguente modo: ad un *Token* corrisponde un solo *User* ed una sola *Resource* associati. Ad uno *User* è associato una sola *Key* e un certo numero di *Token* (0..*).
N.B. Non abbiamo inserito classi database per la lista di risorse, autenticazioni ed autorizzazioni. L’obiettivo è gestirle più comodamente attraverso strutture dati come mappe o arraylist da decidere in fase di implementazione.

Class diagram: specifico Tema comune



Il diagramma in questione copre l'implementazione della comunicazione tra Client e Server e l'elaborazione dei messaggi in arrivo ed in uscita. Non essendo in fase di implementazione saranno inevitabili modifiche al diagramma e alle classi relative.
L'idea di base è la seguente:
Sia Client che Server necessitano di due moduli; uno responsabile della connessione e dell'invio di messaggi(Client/ServerConnectionManager) e l'altro responsabile della ricezione ed elaborazione degli stessi (Client/ServerListener).
N.B. come consigliato nelle specifiche il canale di comunicazione implementato sarà ZeroMQ.

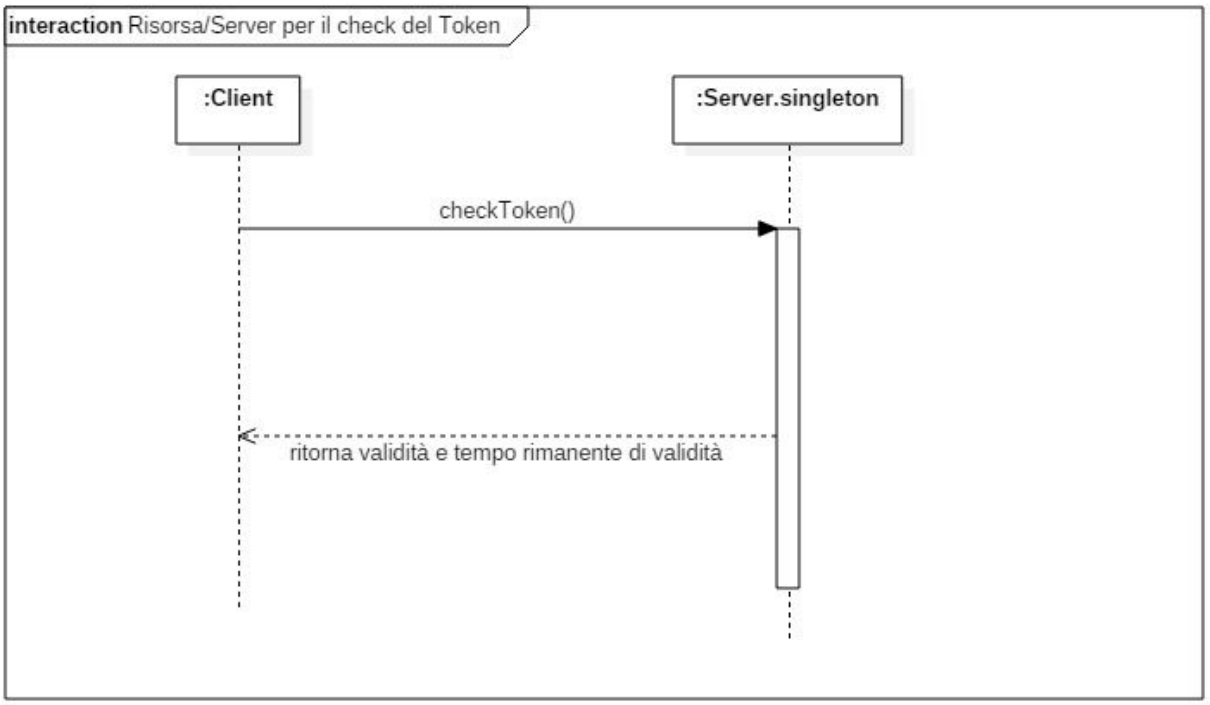
Sequence diagram: autenticazione e autorizzazione Client/Server



L'istanza Client instaura una connessione con il singleton del Server, invocando il metodo getListRes() che restituisce la lista delle risorse accessibili e stampa a video la lista. Al metodo getAuthentication() verrano passati come parametri una scadenza un codice utente e un livello scelto in base al livello relativo alla risorsa per la quale si vuole chiedere l'autorizzazione.
Il metodo crea la chiave, aggiorna la lista di autenticazioni e ritorna al Client la chiave, la quale, una volta memorizzata, verrà passata come parametro insieme al codice della risorsa scelta nel metodo getAuthorization(). Analogamente al metodo autenticazione questo crea il Token, aggiorna la lista delle autorizzazioni e ritorna al Client il Token.

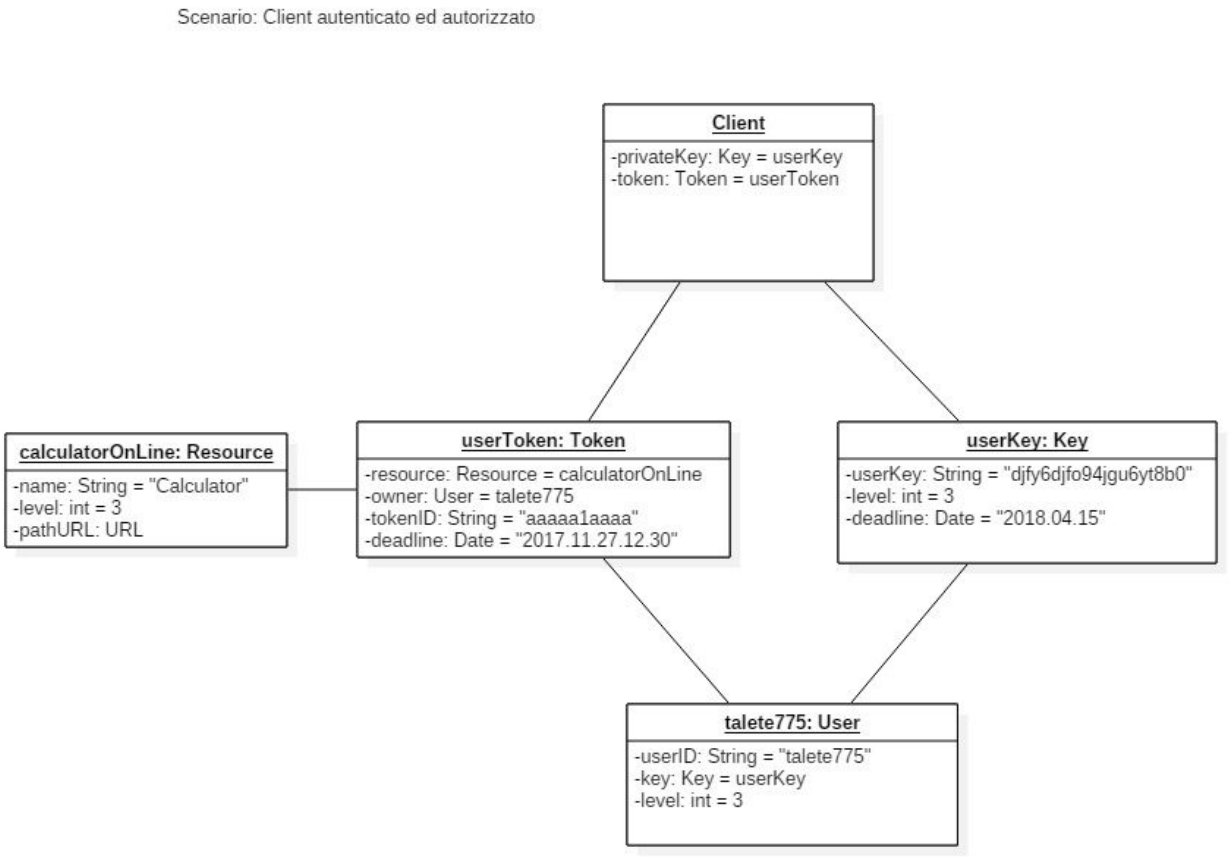
N.B. lo scenario in questione simula la corretta esecuzione delle richieste e risposte; eventuali errori non sono presi in considerazione.
N.B. il metodo getListRes() può essere chiamato in qualsiasi momento dal Client e ritorna la lista di tutte le risorse accessibili, ovvero non implementeremo il parsing della lista in base al livello della chiave dell'utente richiedente la lista.

Sequence diagram: controllo di validità del Token da parte della risorsa



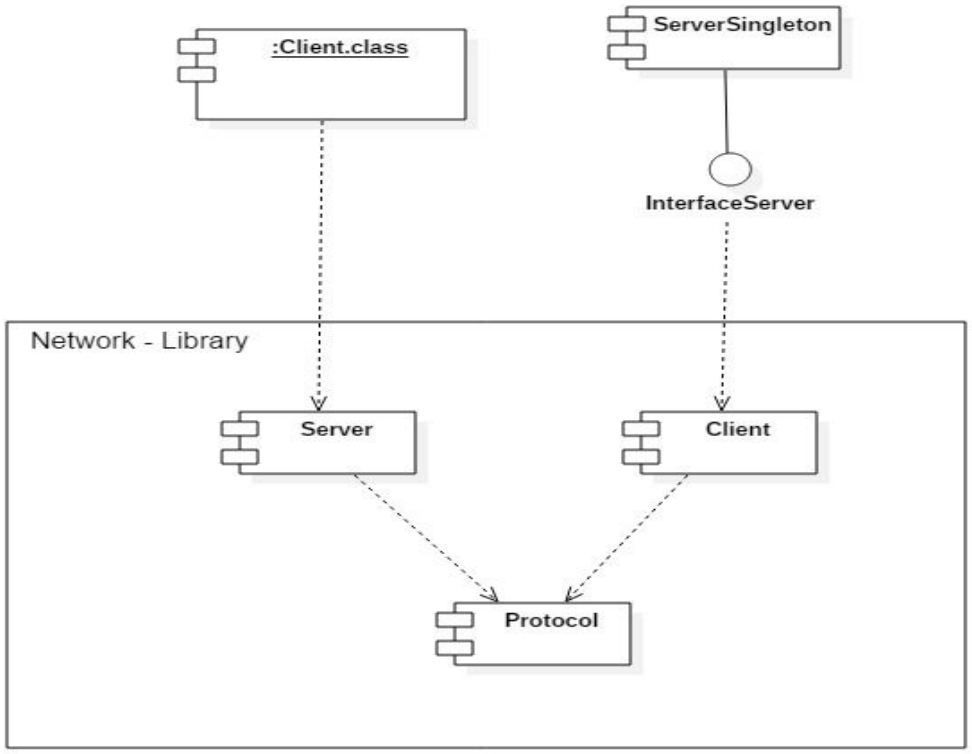
In questo diagramma, si vuole mostrare come un client possa richiedere il controllo di validità di un token. Questo metodo, esposto dal server tramite l'interfaccia *ClientServerInterface*, sarà utilizzato esclusivamente dalle risorse. L'istanza *Client* apre una connessione verso il *Server* chiamando il metodo *checkToken(token)* in modo da avere come risposta l'eventuale conferma di validità del Token o un eventuale messaggio di errore. La conferma di validità del token sarà accompagnata dal numero di minuti rimasti dal suo scadere (il tempo di validità del token è di 24h per specifica).

Object diagram: Client autenticato ed autorizzato



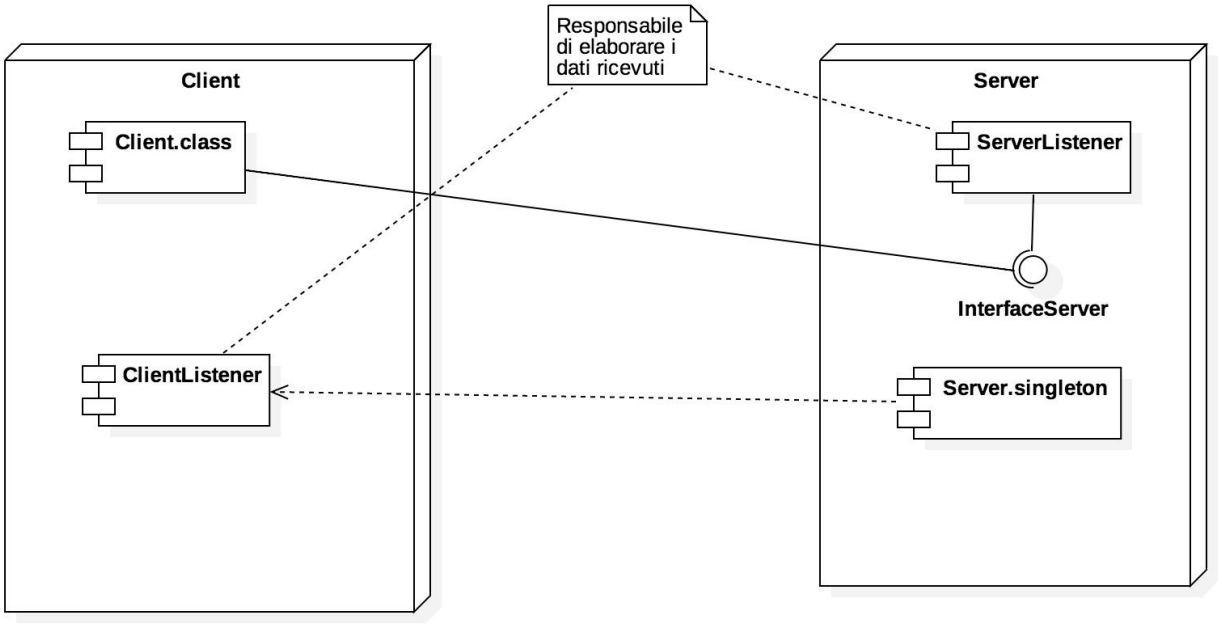
Lo scenario rappresentato è lo stato del Client dopo l'autenticazione e l'autorizzazione, ovvero il Client è in possesso di una chiave associata ad un utente e di un token associato ad una risorsa. N.B. la deadline del token necessita dell'orario in quanto ha validità di 24h.

Component diagram



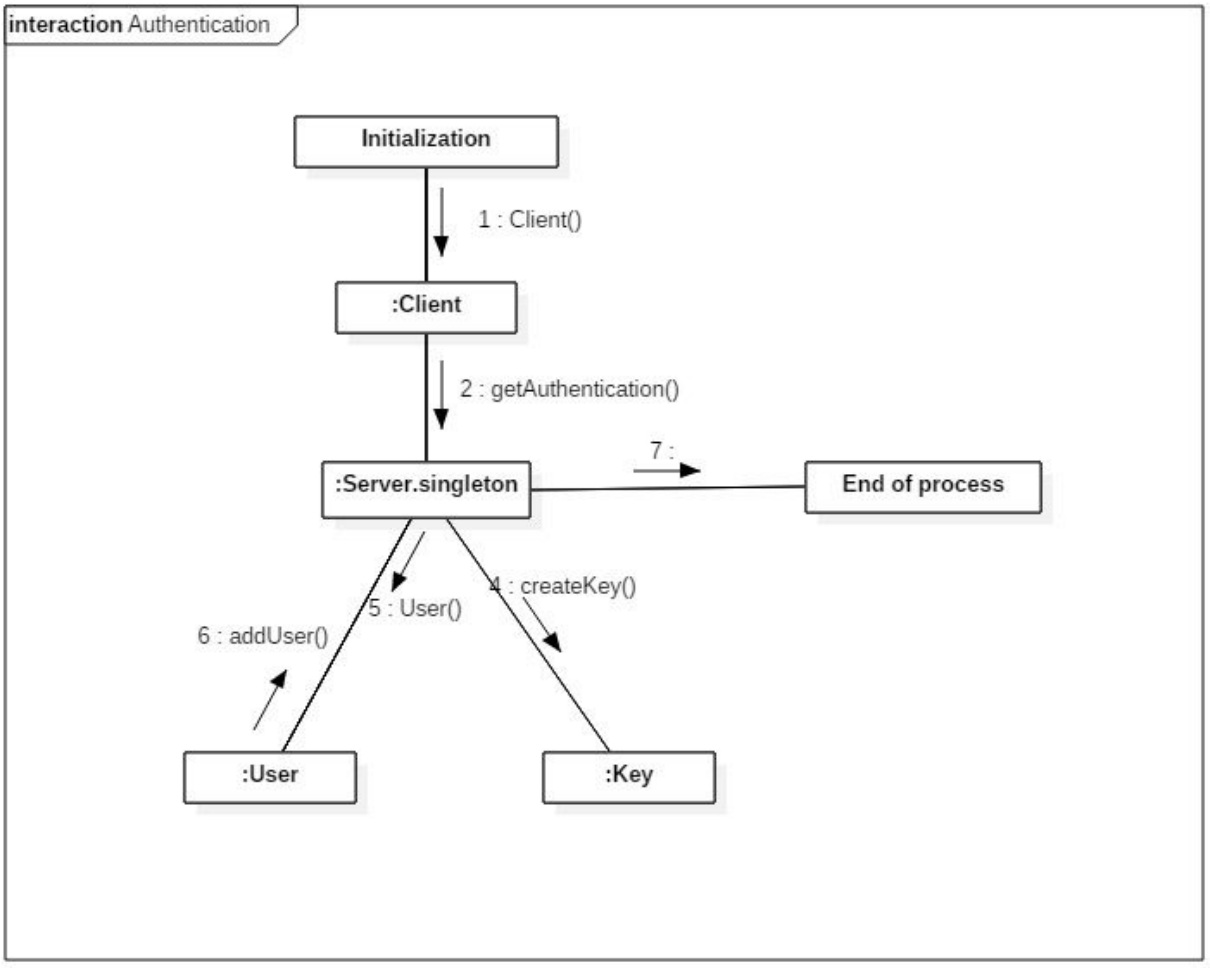
Nel component diagram, sono stati inseriti tutti i componenti che si occupano della trasmissione di dati. Si evince come l'istanza della classe *Client* e l'interfaccia esposta del singleton della classe *Server* utilizzano i componenti di rete offerti dalla libreria. I componenti utilizzati andranno dunque a coprire le responsabilità e le azioni del client e del server. Per maggiori informazioni riguardante la libreria, è possibile consultare il class diagram della libreria.

Deployment diagram



Nel deployment diagram, si evidenziano le dinamiche dei componenti principali durante le varie richieste. Sono presente due risorse computazionali, rispettivamente Server e Client, e i loro componenti che hanno il compito di inviare, ricevere ed elaborare contenuti in formato JSON mediante il protocollo json-rpc.

Collaboration diagram: richiesta di autenticazione



Il diagramma descrive il processo di autenticazione nel quale il server, una volta ricevuta dal Client la RequestObject che chiama il metodo getAuthentication(), in ordine, crea la chiave, la associa ad un'istanza di un utente passandola al costruttore User(), il quale invocherà il metodo, appartenente al server, di aggiornamento della lista di autenticati.

,

Paolo De Santis
Hnia El Jadi
Stefano De Cillis