POLITECNICO
MILANO 1863

Polo territoriale di Como
SCUOLA DI INGEGNERIA INDUSTRIALE E DELL'INFORMAZIONE
INGEGNERIA INFORMATICA
Corso di Ingegneria del Software

AuthOK

PROGETTO DEL CORSO DI INGEGNERIA DEL SOFTWARE
Parte II – Design

Ferrario Stefano
Gumus Tayfun
Isella Paolo
Martinese Federico

Indice

Introduzione

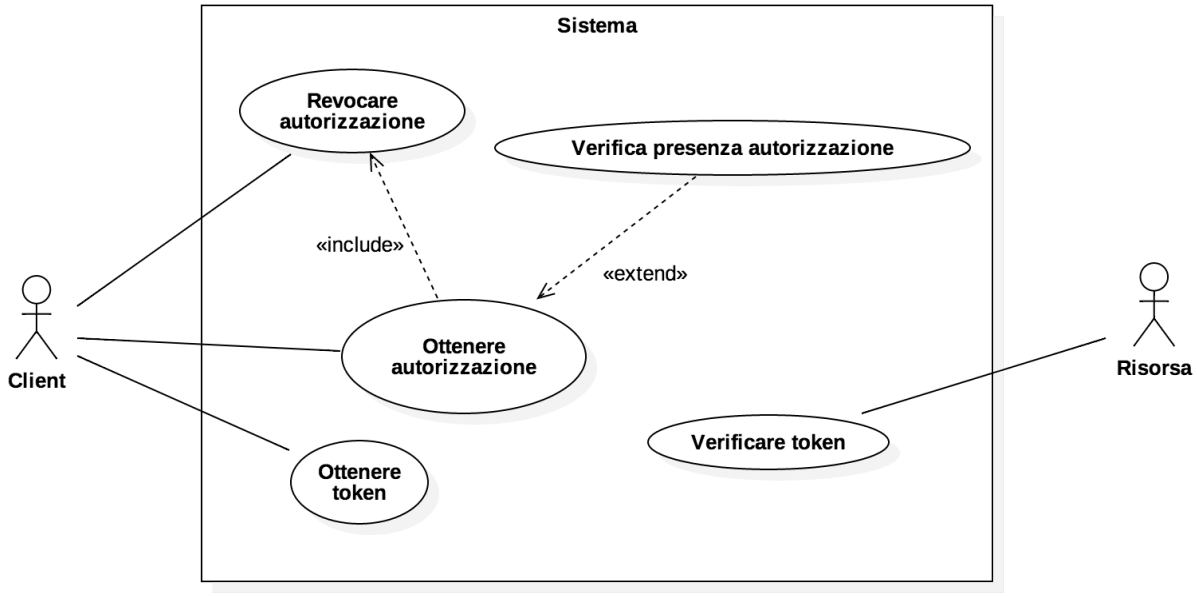
Lo studio del design UML per il progetto AuthOK assume un ruolo di raccordo tra l'analisi dei requisiti e la successiva fase di implementazione. Ciò viene eseguito attraverso la specifica e la documentazione del design dell'intero software. Per svolgere questo passaggio si ricorre all'uso di diversi diagrammi UML. Si è ritenuto opportuno realizzare uno Use Case Diagram, un Class Diagram, due Sequence Diagrams, due Activity Diagrams, due State Diagrams, un Component Diagram, un Deployement Diagram, un Object Diagram ed un Collaboration Diagrams.

Precisazioni

La modellizzazione trattata riguarda la progettazione di un sistema autorizzatore. Verrà omessa la trattazione e la modellizzazione di qualsiasi aspetto riguardante l'accesso ad una risorsa e i passi necessari affinché tale operazione ottenga esito positivo.

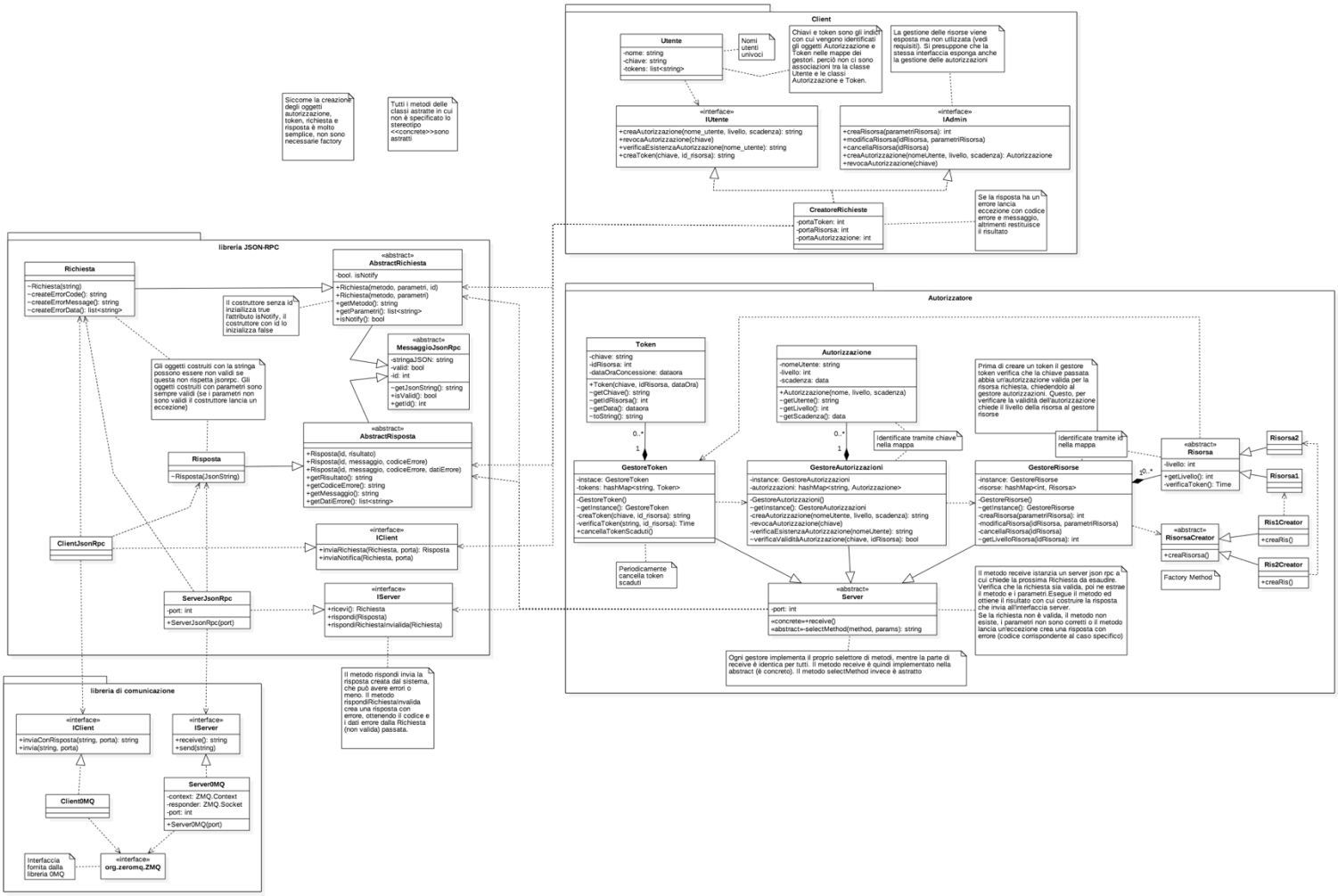
UML - Unified Modeling Language

Use-Case Diagram



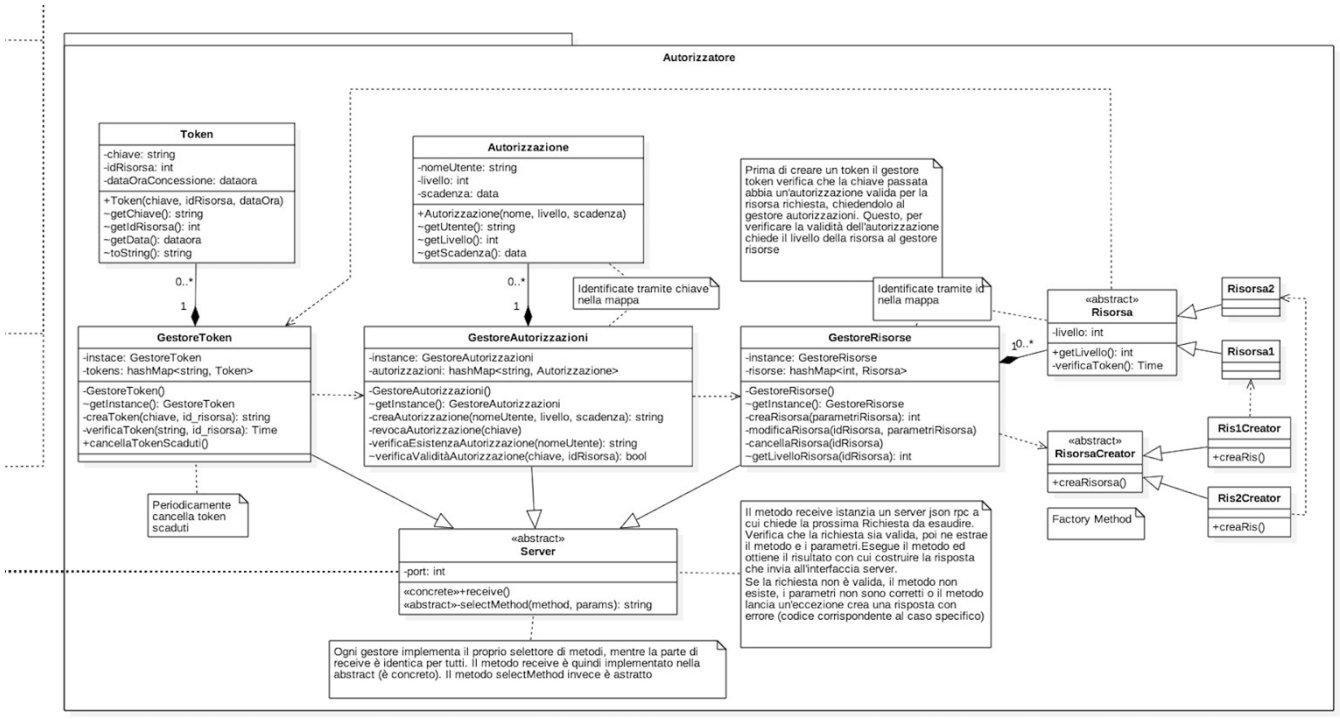
Nel seguente diagramma sono stati scelti quali attori il **client** e la **risorsa**. Entrambi interagiscono con il sistema al fine di raggiungere i loro obiettivi. Il client ha quindi la possibilità di **ottenere un'autorizzazione**. Si ritiene opportuno **verificare la presenza di un'autorizzazione** già associata all'utente preso in considerazione e, in caso di risultato positivo, richiedere all'utente di confermare la volontà di rimuoverla prima di procedere alla creazione di una nuova. Il client potrà inoltre interagire con il sistema al fine di **revocare** una qualsiasi autorizzazione e per **ottenere un token** di accesso ad una risorsa.
Una generica risorsa potrà interagire con il sistema per verificare la validità di un token in qualunque momento.

Class Diagram



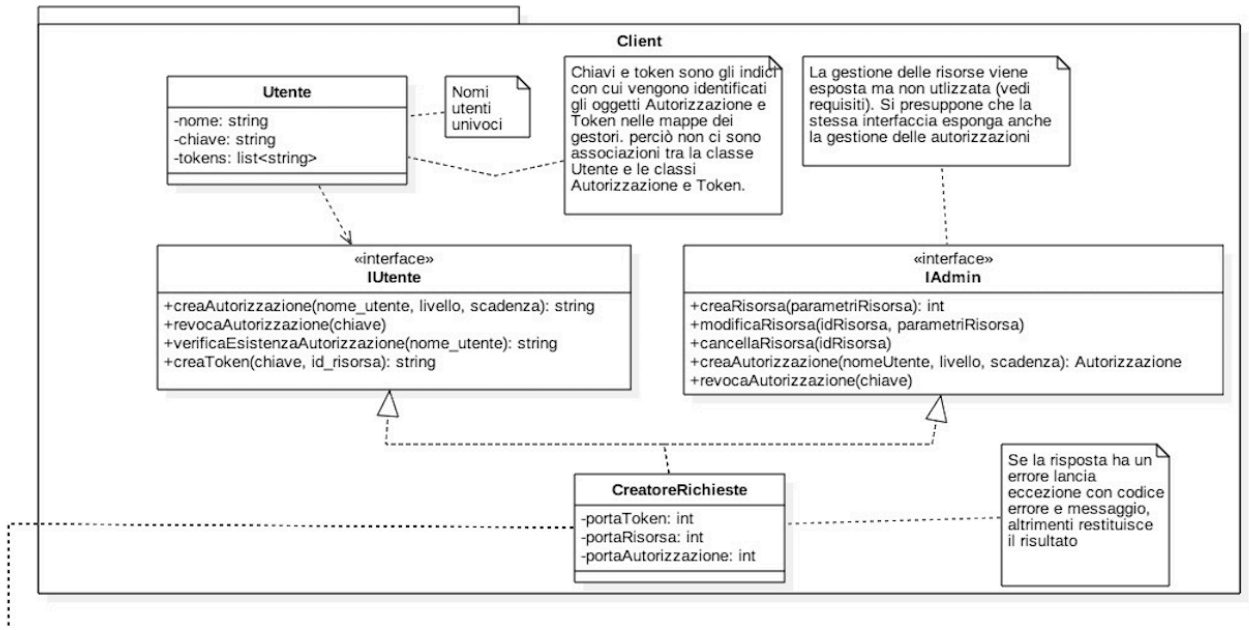
Il sistema si divide in quattro package: **Autorizzatore**, **Client**, **Libreria JSON-RPC** e **Libreria 0MQ**.
Di seguito vengono descritti i singoli package nel dettaglio.

Autorizzatore



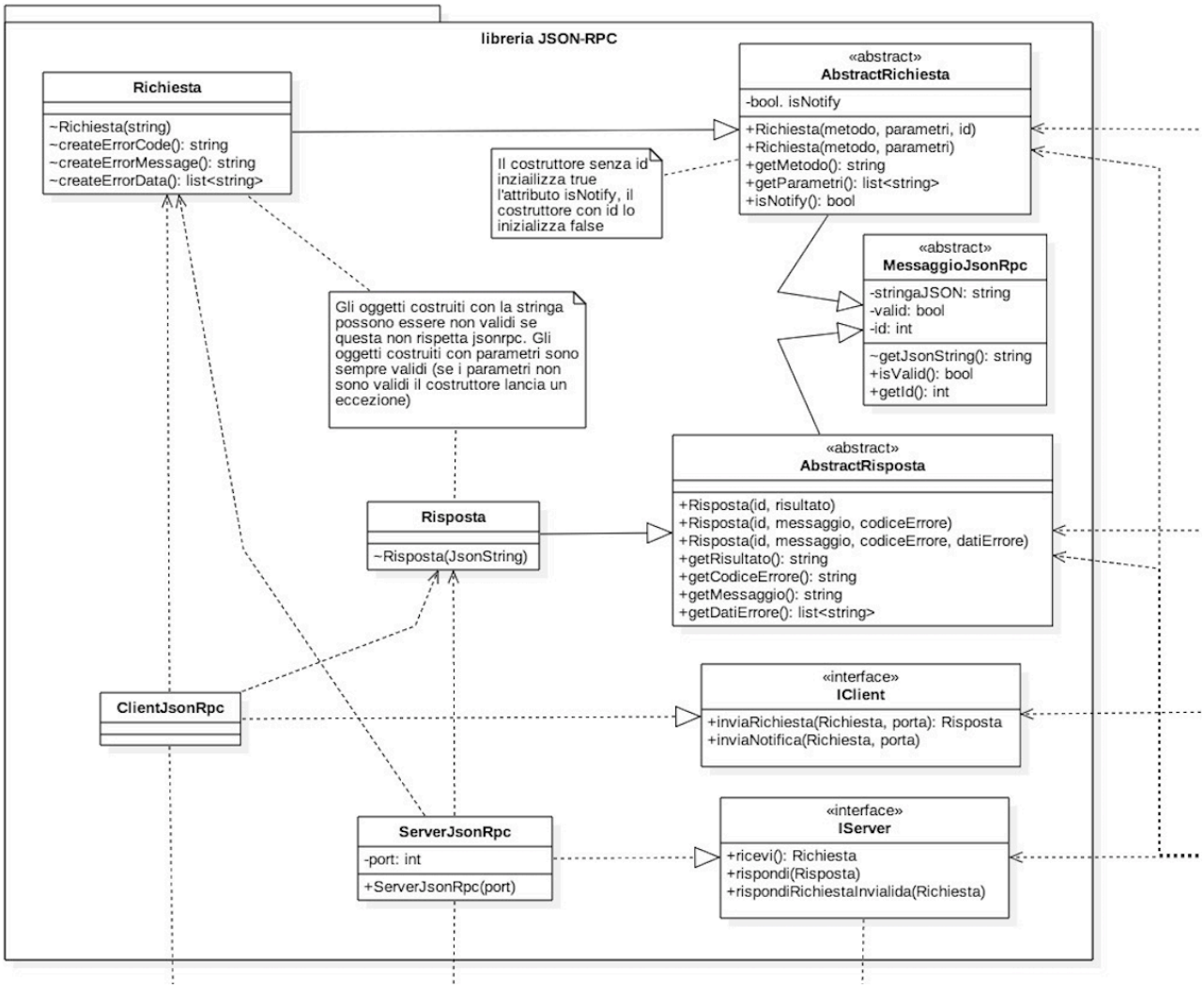
L'autorizzatore è composto dalle classi che identificano i **tre diversi gestori**. Questi ereditano dalla classe astratta "server" che implementa il metodo concreto "receive" e definisce il metodo astratto "selectMethod". I gestori sono **singleton**: per ogni server gestore esiste una sola istanza. Si ricorre a questo design pattern per gestire in maniera univoca i dati (token, autorizzazione e risorsa). Ciascun gestore elaborerà richieste di un determinato tipo compatibilmente al proprio "settore di competenza" e per questo "ascoltano" su porte diverse. Al fine di elaborare le richieste i gestori utilizzano i propri metodi, quali ad esempio "creaToken()" e "revocaAutorizzazione()". I gestori sono inoltre in una relazione di composizione rispettivamente con le classi **Token**, **Autorizzazione** e **Risorsa**. La creazione di token e autorizzazioni è semplice ed implementata nei metodi "creaToken" e "creaAutorizzazione" mentre la creazione risorse è delegata al design pattern del **Factory Method**. Si ricorre a ciò per risolvere il problema della creazione di più risorse di diverso sottotipo.

Client



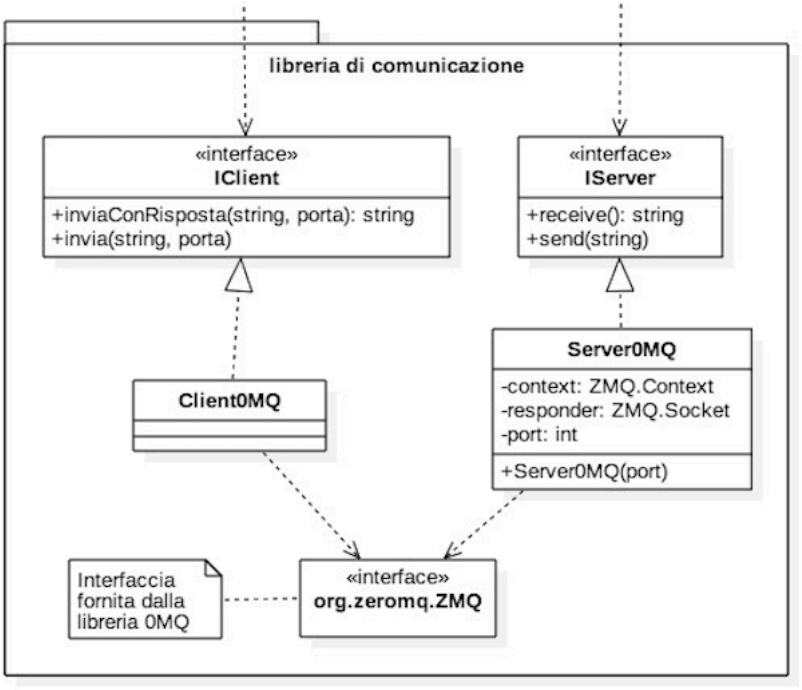
È presente una classe **Utente** che rappresenta gli utenti che utilizzano il sistema. Ogni utente è identificato da un nome univoco e, se autorizzato, memorizza la sua chiave segreta. Inoltre memorizza tutti i token che gli sono stati rilasciati. Chiave e token sono salvati sotto forma di stringa e non come istanze di Token e Autorizzazione. Attraverso l'**interfaccia** predisposta l'utente può chiamare i metodi del **creatore richieste**. Quest'ultimo, in base al metodo invocato, crea una richiesta e invoca l'invio dal client JSON-RPC. Il creatore richieste conosce le porte su cui i gestori sono in ascolto.
Espone inoltre un'interfaccia "admin" che permette la creazione, la modifica e la cancellazione delle risorse. Si ipotizza che la stessa interfaccia esponga la creazione e la revoca delle autorizzazioni. Da specifiche non è previsto l'utilizzo di questa interfaccia.

Libreria JSON-RPC



La libreria JSON-RPC comprende le classi **Richiesta** e **Risposta** che ereditano rispettivamente da *abstract Richiesta* e *abstract Risposta*. Sono classi abstract e non interfacce perché definiscono anche i costruttori. A loro volta tali classi ereditano dalla classe **Messaggio JSON-RPC** anch'essa abstract.
Un messaggio JSON-RPC è una stringa formattata secondo le specifiche del protocollo JSON-RPC e si specifica nelle classi Richiesta e Risposta. Queste possono essere costruite tramite i rispettivi parametri oppure a partire da una stringa appositamente formattata. I costruttori con parametri della richiesta vengono invocati dal creatore richieste (lato Client) mentre quelli della risposta dai gestori (lato Server). I costruttori da stringa vengono invocati all'interno della libreria. Il **client JSON-RPC** espone i metodi necessari per l'invio di richieste o notifiche che implementa sfruttando una libreria di comunicazioni (in questo caso la Libreria 0MQ). Ricevuta una stringa come risposta e la utilizza per ricreare l'oggetto risposta inviato dal server. Allo stesso modo, il **server JSON-RPC** riceve stringhe che utilizza per ricostruire la richiesta che è stata inviata dal client a cui risponde con "oggetti risposta".

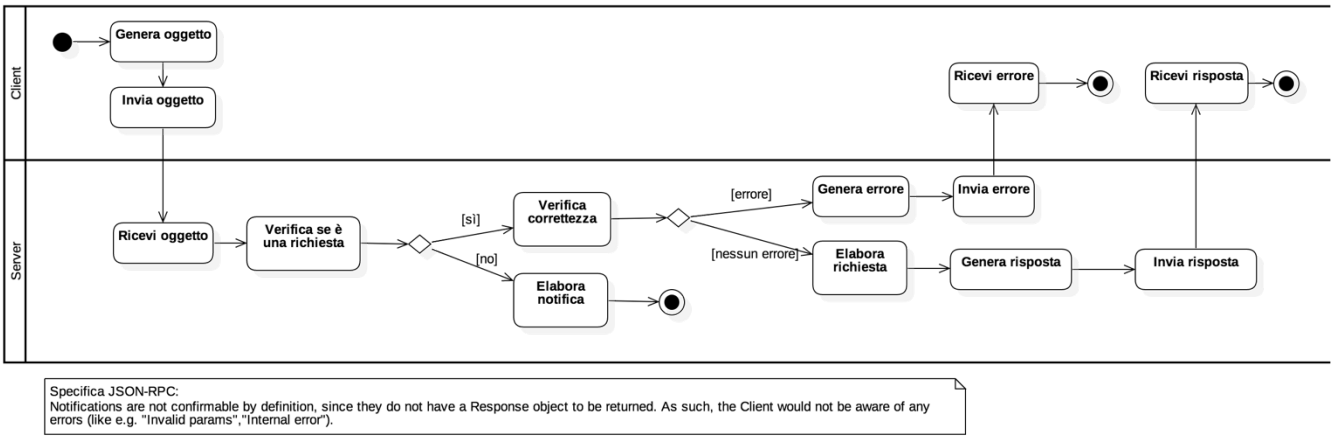
Libreria di comunicazione



Presenta due classi: **Client 0MQ** e **Server 0MQ** che utilizzano la **libreria 0MQ**. Il client invia stringhe ad una porta e ottiene delle stringhe di risposta, il server ascolta su una porta e riceve stringhe a cui rispondere.

Activity Diagram

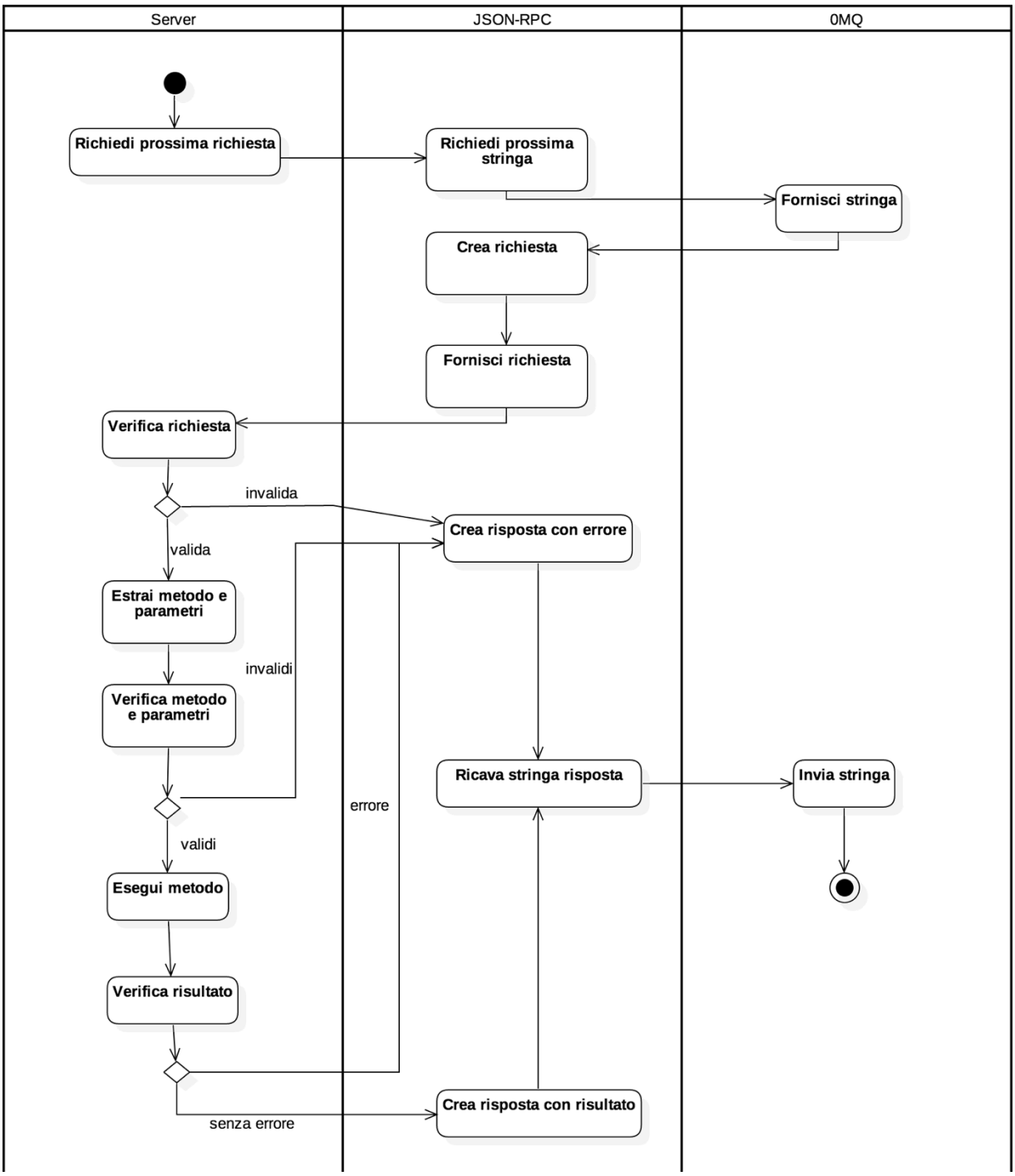
Activity Diagram – Tema comune



L'Activity Diagram riferito al tema comune del progetto riguarda la comunicazione client server necessaria per il funzionamento del sistema di autorizzazione. Si mostra che, alla **generazione di un oggetto** e al successivo **invio** da parte del client, segue una **verifica della richiesta** ricevuta da parte del server. Se l'oggetto ricevuto è una richiesta viene **verificata la correttezza** delle informazioni in essa contenute, in caso contrario si tratterà di una notifica che dovrà essere elaborata. In base al risultato della verifica si procederà alla **generazione di un errore** o all'**elaborazione della richiesta** e quindi alla **generazione della risposta**.

Le notifiche non generano errori ed il client non ha modo di sapere di aver inviato una notifica errata.

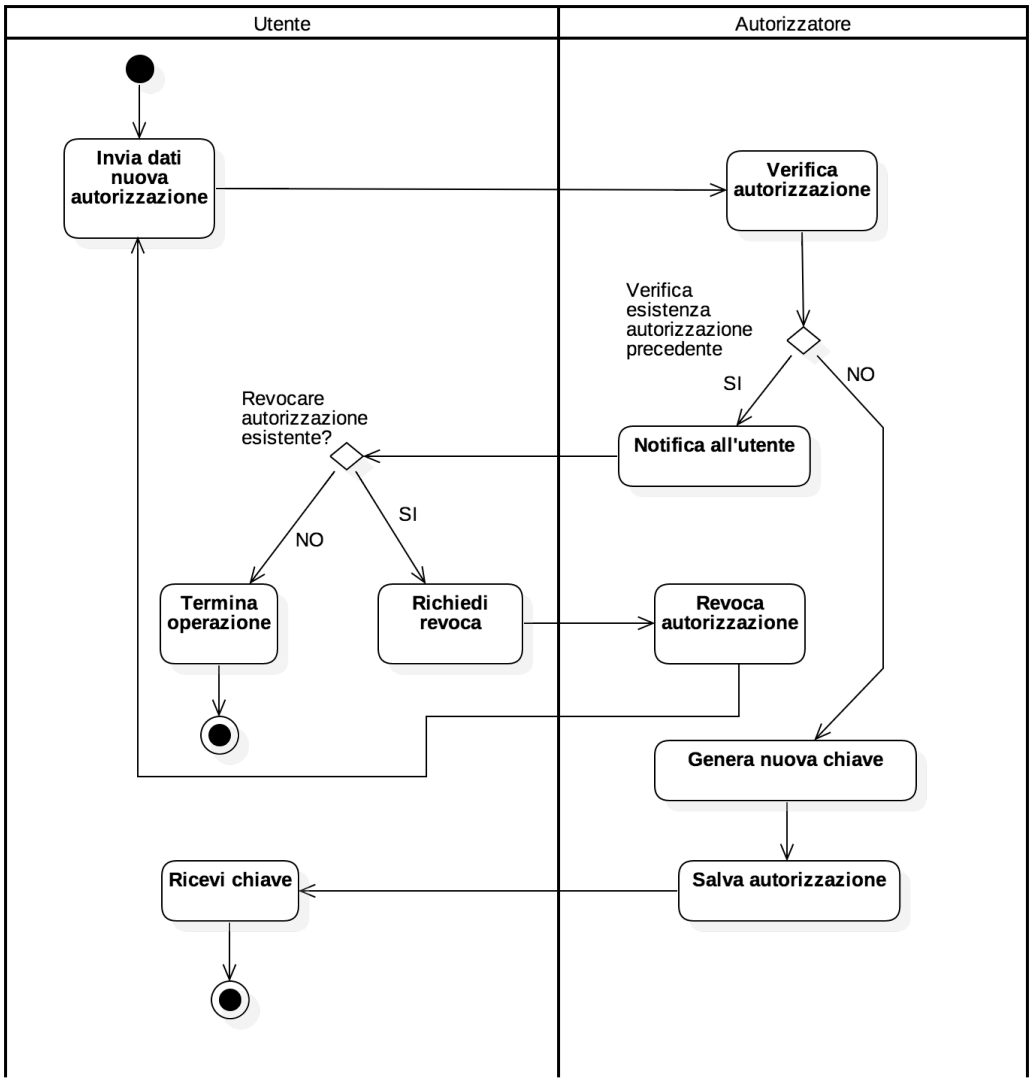
Activity Diagram – Server



Nel diagramma seguente viene mostrato il processo di gestione delle richieste da parte del server e i passaggi che richiedono l'utilizzo delle librerie JSON-RPC e 0MQ
Si assume che il server rimanga in attesa di nuove richieste da parte del client. A sua volta anche il server JSON-RPC attenderà la disponibilità di una nuova stringa da elaborare che invierà alla libreria 0MQ. Tale libreria ha il ruolo di canale di comunicazione tra il server e il client JSON-RPC.
La creazione di una nuova richiesta richiederà una **verifica da parte del Server** che in caso di richiesta non valida **restituirà errore**, altrimenti procederà all'**estrazione del metodo e dei**

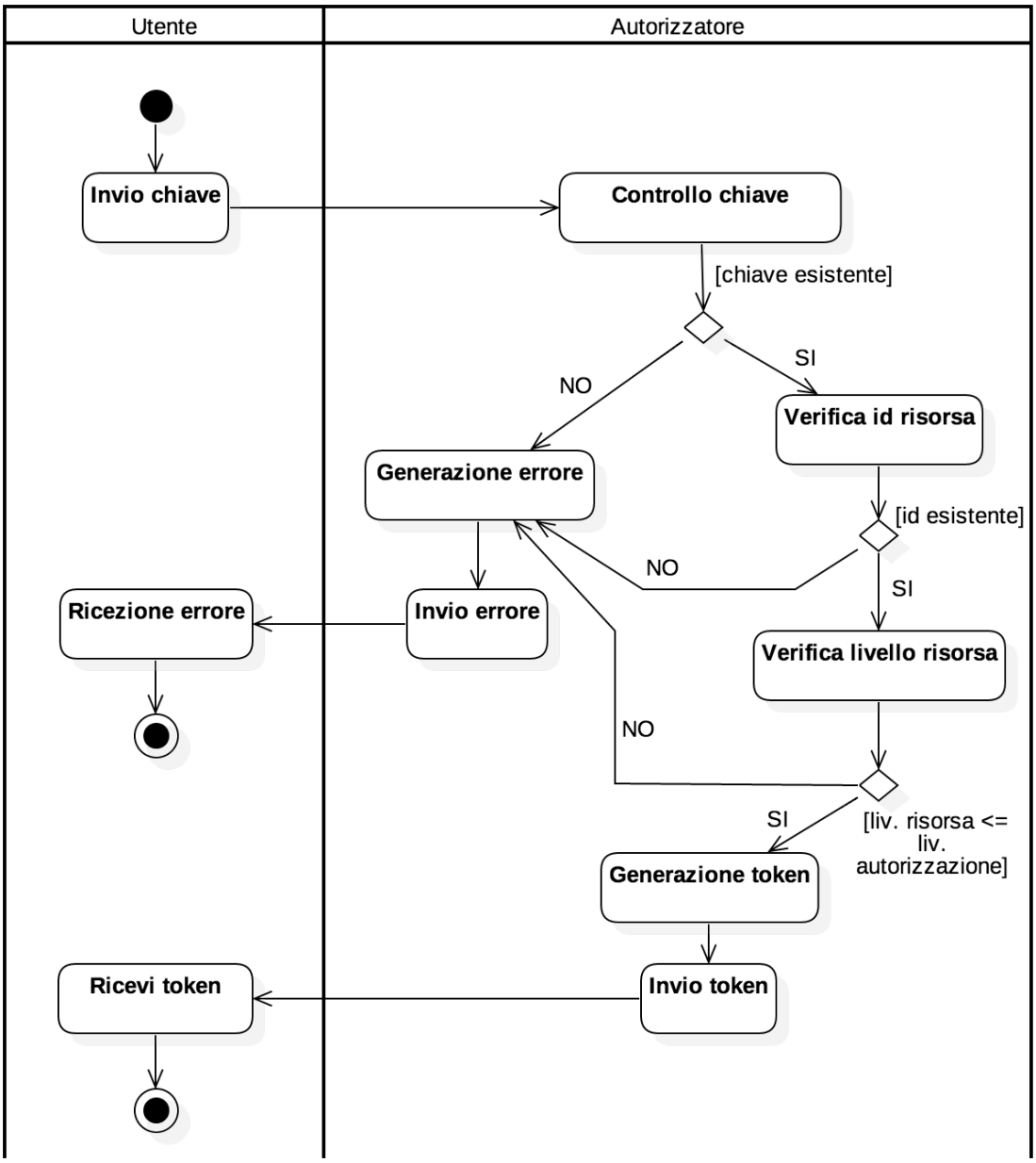
parametri. Dopo una successiva verifica di questi, il sistema potrà **riportare un errore** oppure **eseguire il metodo** di cui è stata richiesta l'invocazione.
Ancora una volta, una **verifica del risultato**, potrà riscontrare errori oppure portare alla **creazione di una risposta** con il risultato ottenuto.
Qualsiasi risposta, sia essa un errore o contente un risultato finale di un'operazione, verrà trasformata in stringa JSON-RPC per poter procedere alla ritrasmissione all'utente.

Activity Diagram – Creazione autorizzazione



Si ritiene significativo mostrare con il seguente diagramma i passaggi eseguiti dall'utente e dall'autorizzatore nel processo di creazione di una nuova chiave.
In primo luogo, viene richiesto all'utente l'**inserimento dei dati necessari** per la richiesta dell'autorizzazione. Il sistema, che **riceve la richiesta**, **effettua una verifica** e qualora sia presente una precedente autorizzazione riferita all'utente ne **richiederà la revoca** prima di poter proseguire con la creazione di una nuova.
Se non sono presenti autorizzazioni relative all'utente che si vuole abilitare il sistema procede alla **generazione di una nuova chiave** e al **salvataggio dell'autorizzazione** appena creata. Procederà quindi con l'**invio della chiave** all'utente.

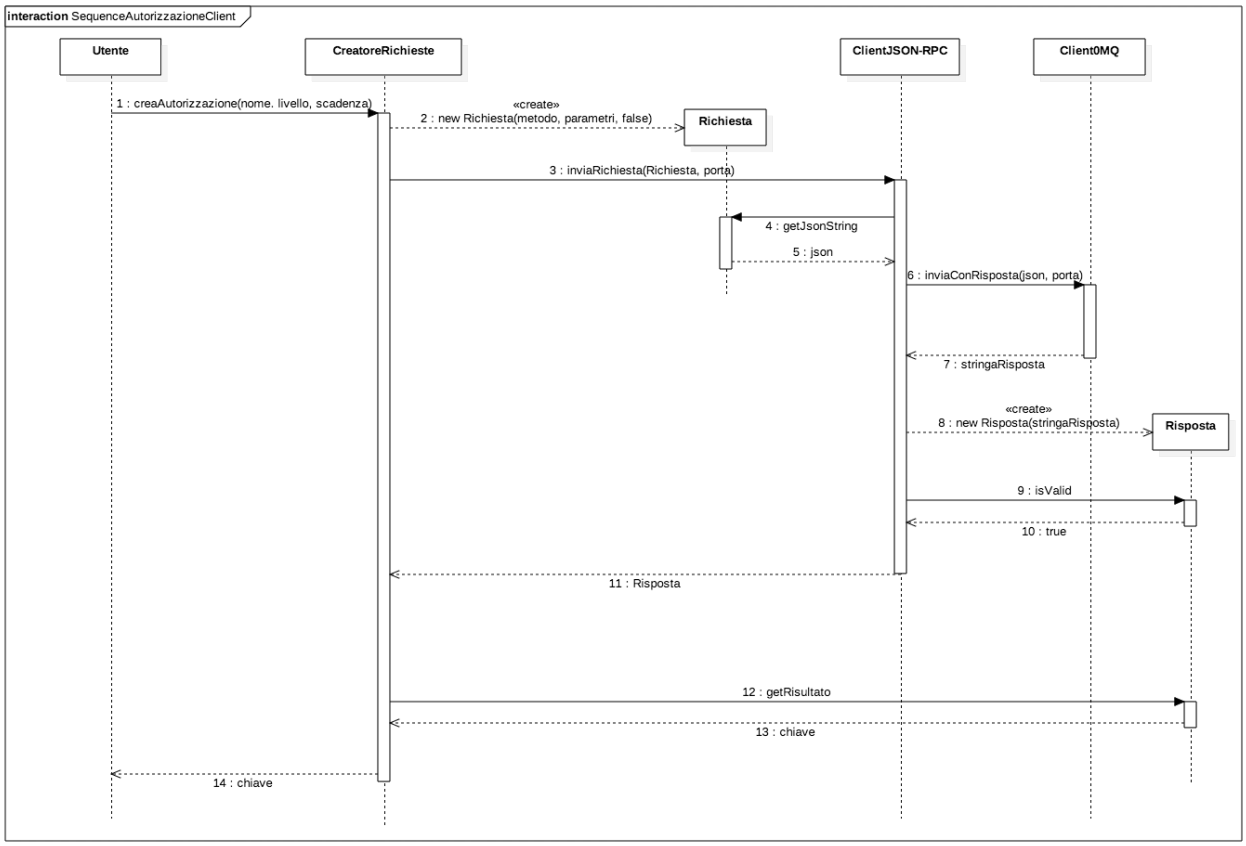
Activity Diagram – Creazione token



Nel seguente diagramma vengono mostrati i passaggi effettuati da utente e autorizzatore per la creazione di un token necessario all'utente per accedere ad una risorsa.
Per dare inizio alla creazione l'utente **invia all'autorizzatore** la chiave in suo possesso.
L'autorizzatore procede alla **verifica della chiave** e, in caso di non validità **genera ed invia un errore all'utente**. Qualora la chiave fornita sia valida l'autorizzatore **eseguirà un'ulteriore verifica sull'ID della risorsa**, fornito dall'utente. Se l'operazione non restituisce errore si procede con l'ultima verifica che prevede il **controllo del livello** a cui si vuole accedere. Se tutte le verifiche vengono superate l'autorizzatore **genera il token** che invierà all'utente.

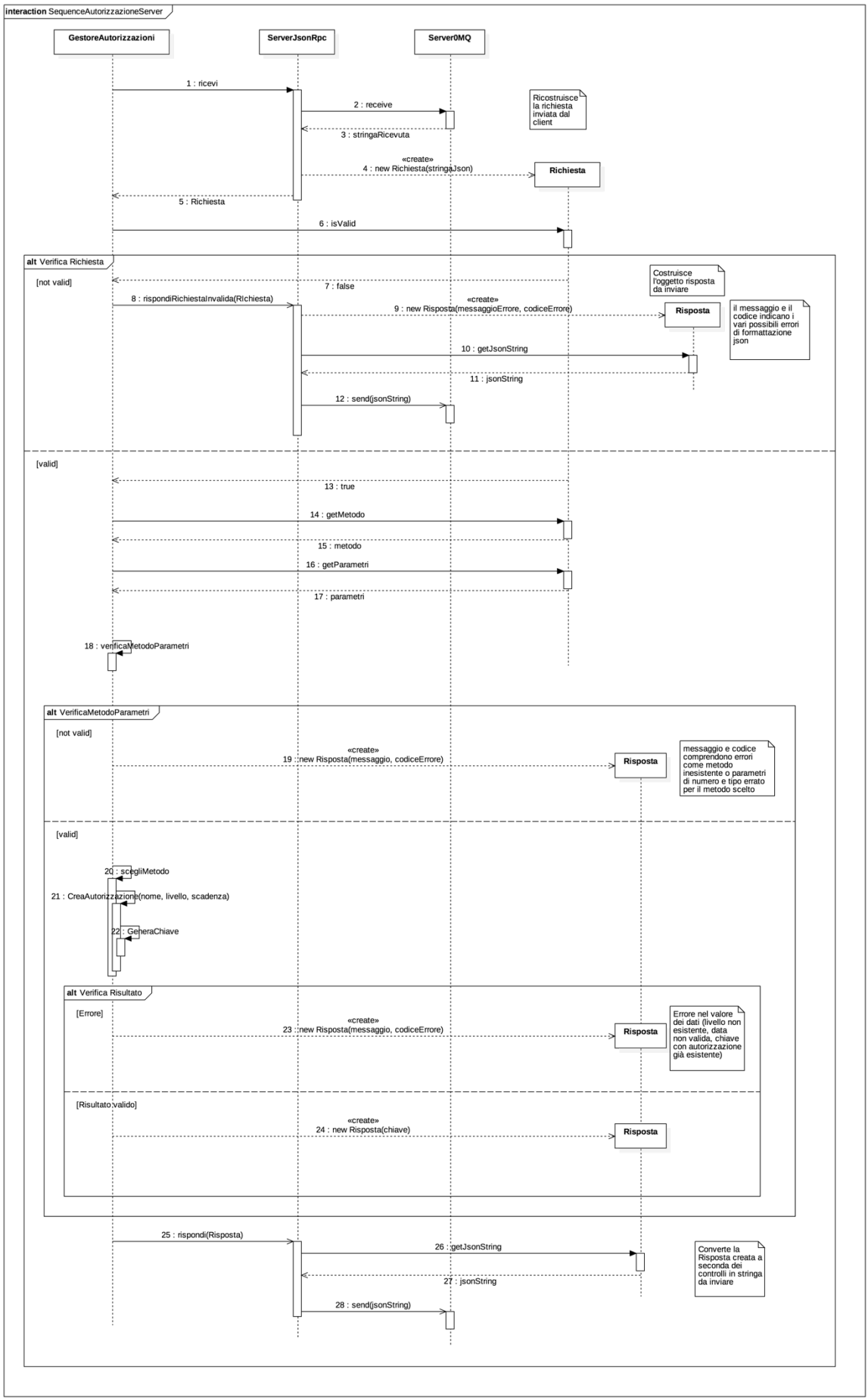
Sequence Diagram

Sequence Diagram – Autorizzazione Client



Nel seguente diagramma vengono descritti in sequenza i passaggi eseguiti da **utente**, **creatore richieste**, **client JSON-RPC** e **client 0MQ** al fine di richiedere una nuova autorizzazione e quindi, ottenere una chiave. All'inserimento dei dati da parte dell'utente per la creazione di una nuova autorizzazione, il creatore richieste istanzia una **nuova richiesta**. I dati inseriti vengono inviati al client JSON-RPC che otterrà poi, attraverso una rielaborazione dei dati, una stringa. Mediante l'interazione con il client 0MQ si ottiene una stringa di risposta che porterà il client JSON-RPC ad istanziare una **nuova risposta**. Dopo averne verificato la validità, si restituisce la risposta al creatore richieste che gestirà la presenza di errori ed il risultato della risposta che, in questo caso, è la chiave. Terminati questi passaggi viene restituita la chiave all'utente.
In questo diagramma si è deciso di mostrare una situazione nel quale ogni passaggio va a buon fine e non vengono riscontrati errori.

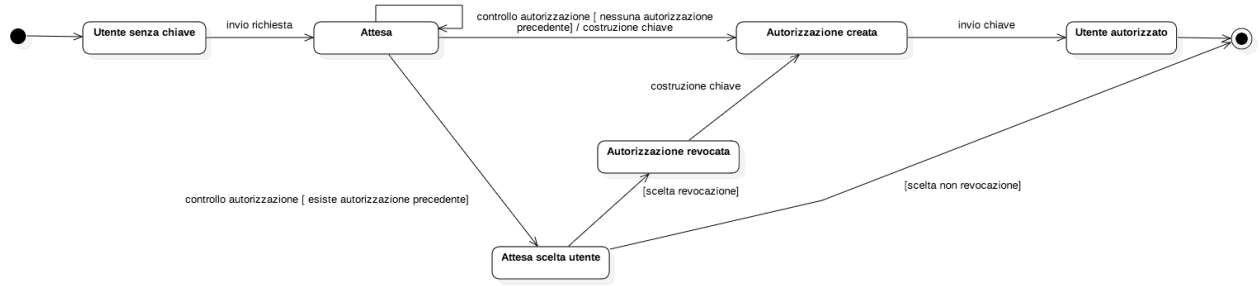
Sequence Diagram – Autorizzazione Server



Nel seguente diagramma si descrive più nel dettaglio i passaggi eseguiti dal lato server. Considereremo quindi i metodi richiamati dal gestore autorizzazioni, dal server JSON-RPC, dal server 0MQ, dalla richiesta e dalla risposta. Il gestore autorizzazioni rimane in attesa di ricevere nuove richieste dal server JSON-RPC il quale, allo stesso modo, attenderà nuove richieste dal server 0MQ.
In presenza di una nuova richiesta ne **verifica la validità**. Distinguiamo quindi i casi "**richiesta non valida**", caratterizzato dalla restituzione dell'errore al gestore autorizzazioni, e il caso "**richiesta valida**" che prevede l'analisi della stringa e una seconda verifica.
In caso di "**non validità**" verrà creata una risposta che **segnalerà errore** altrimenti si procederà alla **creazione di una nuova autorizzazione** e quindi alla generazione della chiave. Nel processo di creazione della chiave il gestore autorizzazioni potrà riscontrare un problema nei dati forniti (*Esempio: Livello non disponibile*) e **creare una risposta contenete il codice dell'errore** oppure, nel caso in cui il processo vada a buon fine, **creare una risposta contenente la chiave stessa**.
Qualsiasi risposta viene convertita in stringa mediante il metodo "getJSON-RPCString" che ne consentirà la trasmissione all'utente come indicato nel primo Sequence Diagram riportato a pagina 9.
La chiamata 25 "rispondi" viene effettuata utilizzando come parametro l'oggetto Risposta che è stato creato in uno dei tre possibili rami. Lo stesso oggetto sarà quello invocato dalla chiamata 26 getJSON-RPCString e che restituisce la risposta 27 JSON-RPCString. Durante l'esecuzione uno solo dei rami verrà seguito e di conseguenza ci sarà una sola istanza di Risposta da inviare e quindi da cui ricavare la stringa JSON-RPC.

State Diagram

State Diagram – Autorizzazione

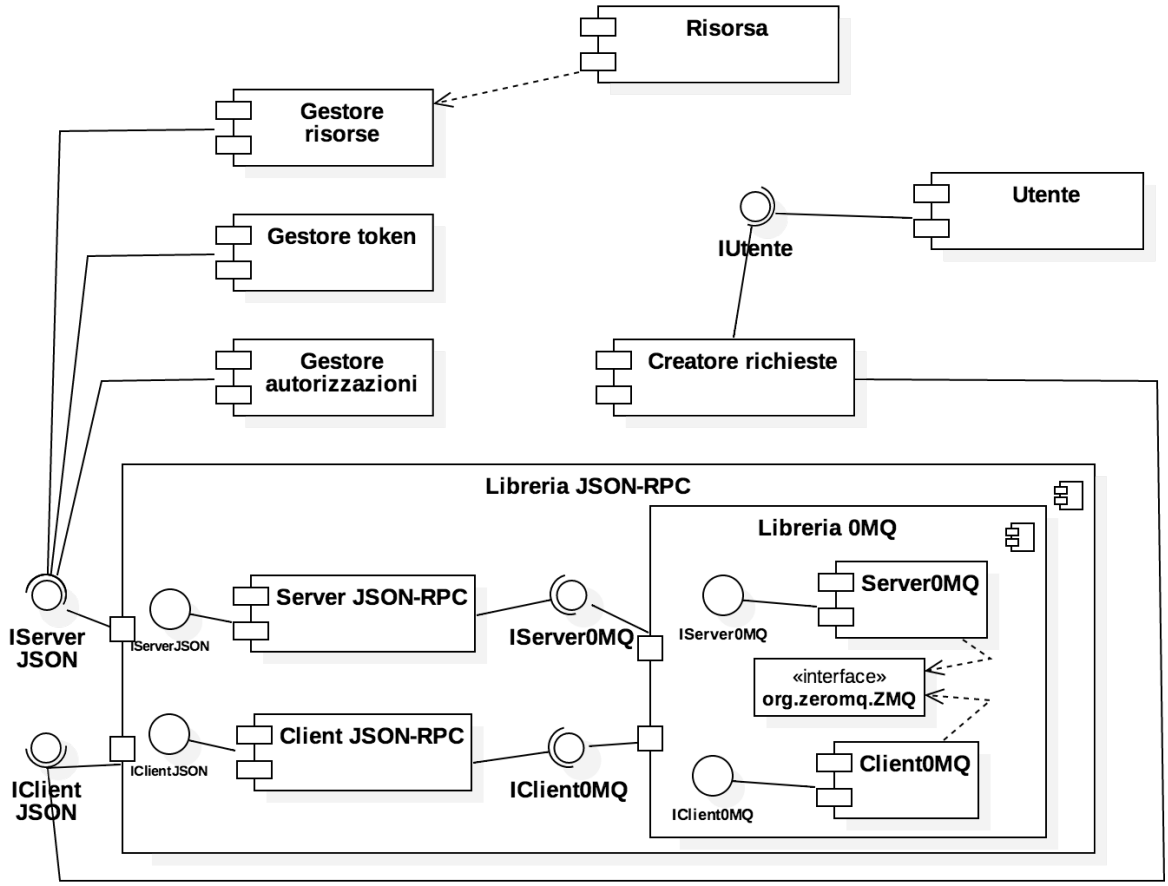


Nel seguente diagramma si rappresentano i diversi stati durante il processo di richiesta di una nuova autorizzazione. Si identificano uno stato "**utente senza chiave**" che attraverso l'invio di una richiesta per una nuova autorizzazione porta ad uno stato "**attesa**". Qui si giunge allo stato "**autorizzazione creata**" e quindi, dopo l'invio della chiave allo stato "**utente autorizzato**". Se viene rilevata la presenza di un'autorizzazione già esistente si giunge ad uno stato che attende la scelta dell'utente per poter procedere all'eliminazione di un'autorizzazione esistente. In base alla scelta si passerà ad uno stato "**autorizzazione revocata**" e quindi alla creazione di una nuova autorizzazione oppure allo stato finale.

State Diagram – Token

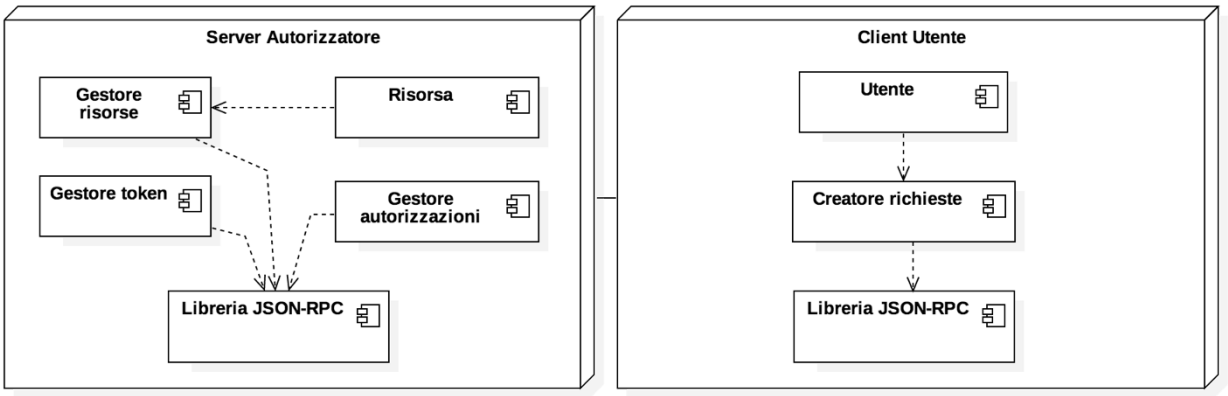
In questo state diagram vengono rappresentati gli stati che caratterizzano un processo di richiesta di un token per l'accesso ad una risorsa. Partendo dallo stato "**utente senza token**", dopo l'invio di una richiesta si giunge ad uno stato di elaborazione della richiesta. A questo punto si **verificherà la validità dell'autorizzazione e che il livello dell'autorizzazione sia maggiore o uguale a quello della risorsa**. Se la richiesta viene respinta si giunge ad uno stato finale senza che alcun token venga fornito, altrimenti si procede fornendo il token all'utente e processo termina dopo lo stato "**utente con token**".

Component Diagram



Nel seguente diagramma vengono descritti i principali componenti del sistema di autorizzazione AuthOK. L'utente attraverso un'interfaccia apposita comunica con il creatore richieste. La libreria JSON-RPC costituisce un importante componente del sistema in quanto consente la comunicazione tra la "parte utente" del sistema e la "parte autorizzatore". Al suo interno sono presenti un componente server e un componente client che a loro volta comunicano grazie alla libreria 0MQ che svolge il ruolo di canale di comunicazione.
Attraverso questi passaggi le richieste giungono ai diversi gestori (gestore autorizzazioni, gestore token, gestore risorse) che le elaborano e ne portano a termine gli obiettivi.

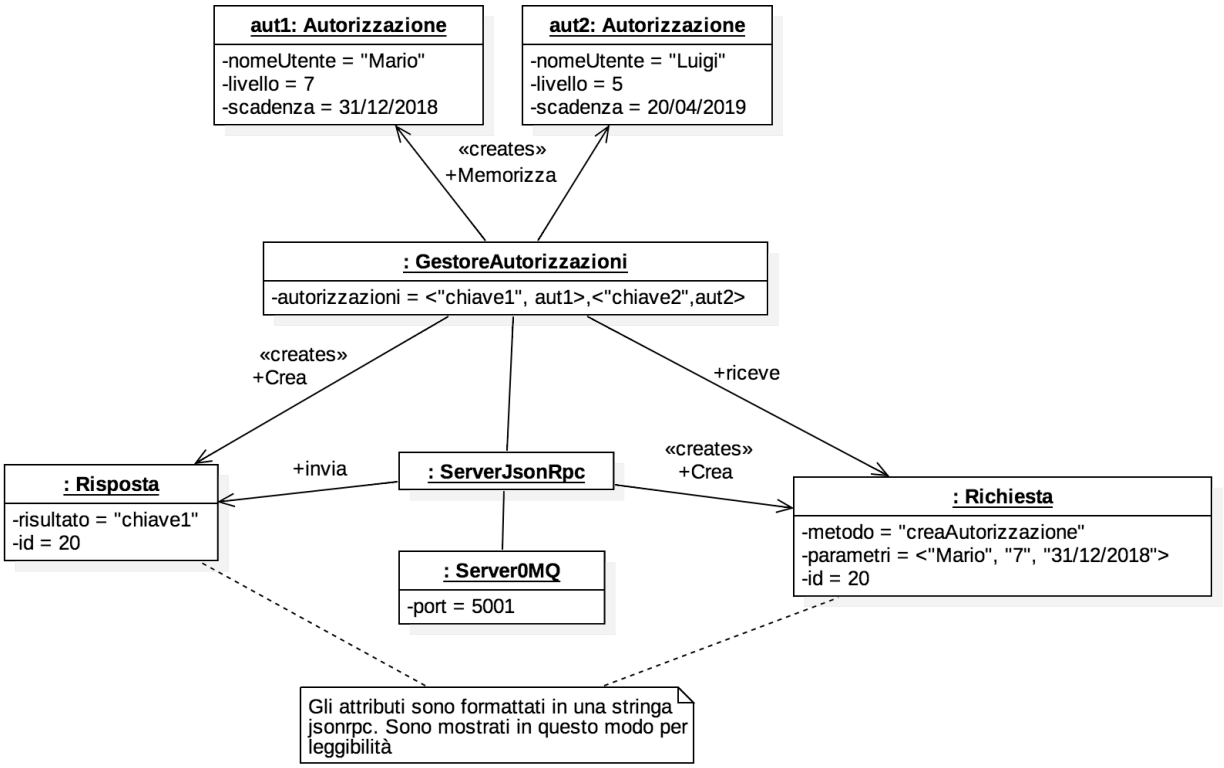
Deployment Diagram



Attraverso la rappresentazione di un Deployment Diagram i diversi componenti precedentemente mostrati vengono inseriti nei due nodi che caratterizzano il sistema di autorizzazione AuthOK. Come si può notare il componente riguardante la libreria JSON-RPC viene inserito e sviluppato in entrambe i nodi in quanto svolge un ruolo centrale nella comunicazione delle due parti. Come specificato nel Component Diagram (pagina 15) la libreria JSON-RPC contiene i componenti client, server e la libreria 0MQ.

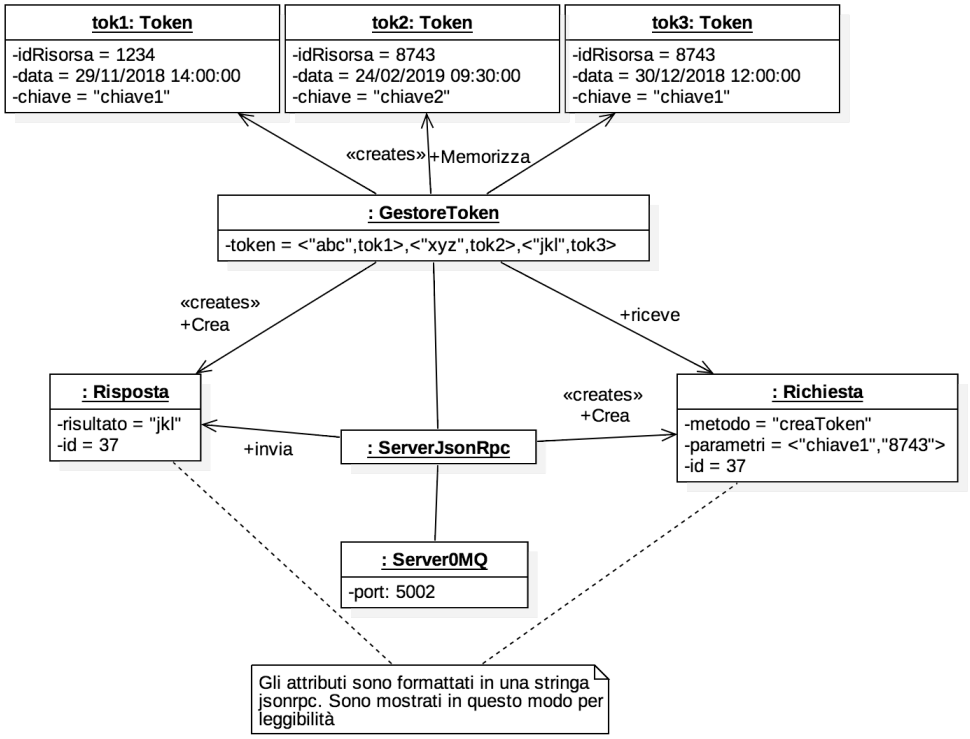
Object Diagram

Object Diagram – Autorizzazione



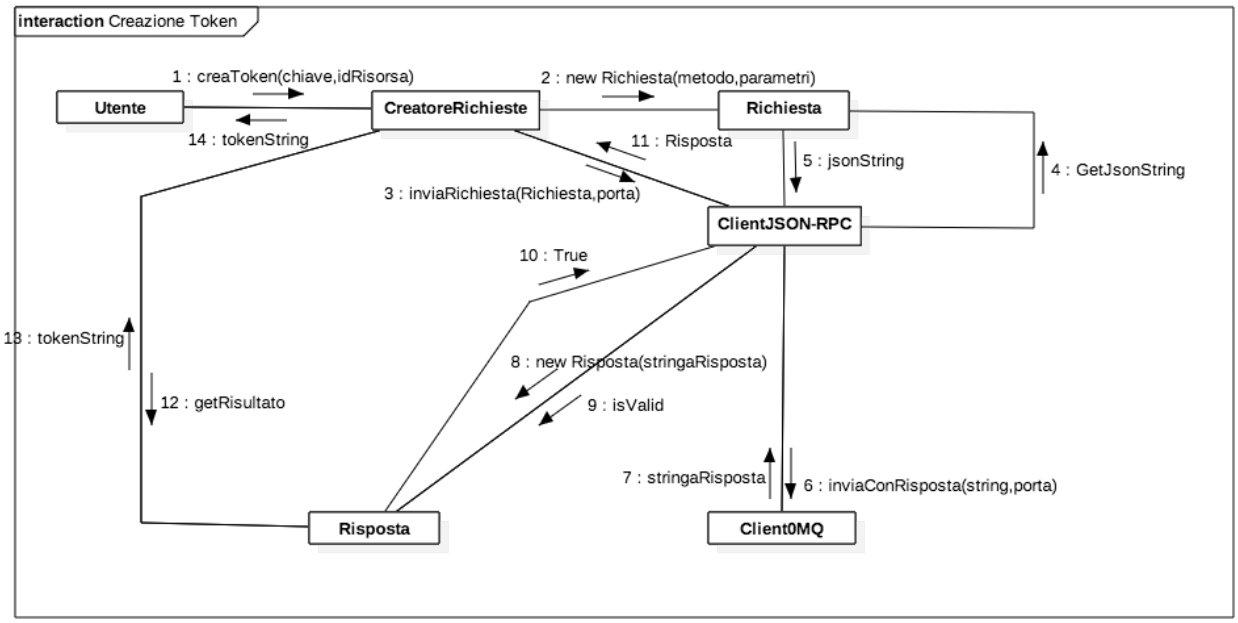
L'Object Diagram riportato offre una rappresentazione statica del sistema nel momento della creazione di una nuova autorizzazione. Il "ServerJsonRpc" genera una "Richiesta" che contiene il metodo "creaAutorizzazione", i parametri e un'ID. Questa istanza viene ricevuta dal "GestoreAutrorizzazioni" che procede con la creazione di una nuova autorizzazione che memorizza. La chiave viene poi restituita all'utente mediante la creazione di una "Risposta", poi inviata dal "ServerJsonRpc".

Object Diagram – Token



In modo del tutto analogo al caso dell'Autorizzazione, in questo Object Diagram viene si ha una rappresentazione statica del sistema durante il processo di creazione di un nuovo token. Anche in questo caso il "ServerJsonRpc" istanzia una nuova "Richiesta" contenente il metodo, in questo caso "creaToken", i parametri e l'ID. Il "GestoreToken" riceve la richiesta e crea il token che memorizza. Il token viene restituito all'utente mediante la creazione di una "Risposta", restituita all'utente dal "ServerJsonRpc".

Collaboration Diagram



Nel seguente collaboration diagram si rappresenta il processo di creazione token. L'utente richiede la creazione di un nuovo token mediante il metodo creaToken(). Viene pertanto creata e inviata una nuova richiesta che viene successivamente rielaborata in forma di stringa. La richiesta viene inviata al client0MQ che interagendo poi con il "lato server" restituisce al Client JSON-RPC la risposta. Il creatore richieste è in grado di ricevere come risposta il risultato ottenuto dall'operazione invocata, in questo caso il token.

Identificativo gruppo: 10
Progetto: Tema A
Componenti:

845386	Ferrario Stefano	stefano6.ferrario@mail.polimi.it
843994	Gumus Tayfun	tayfun.gumus@mail.polimi.it
854412	Isella Paolo	paolo.isella@mail.polimi.it
845326	Martinese Federico	federico.martinese@mail.polimi.it