

*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 3](#_Toc498975062)

[Precisazioni 3](#_Toc498975063)

[UML - Unified Modeling Language 4](#_Toc498975064)

[Use-Case Diagram 4](#_Toc498975065)

[Class Diagram 5](#_Toc498975066)

[Activity Diagram 6](#_Toc498975067)

[**Activity Diagram – Server** 6](#_Toc498975068)

[**Activity Diagram – Creazione autorizzazione** 8](#_Toc498975069)

[Sequence Diagram 9](#_Toc498975070)

[**Sequence Diagram – Autorizzazione Client** 9](#_Toc498975071)

[**Sequence Diagram – Autorizzazione Server** 10](#_Toc498975072)

[State Diagram 12](#_Toc498975073)

[**State Diagram – Autorizzazione client** 12](#_Toc498975074)

[Component Diagram 14](#_Toc498975075)

[Deployment Diagram 15](#_Toc498975076)

[Object Diagram 16](#_Toc498975077)

[Collaboration Diagram 17](#_Toc498975078)

**Introduzione**

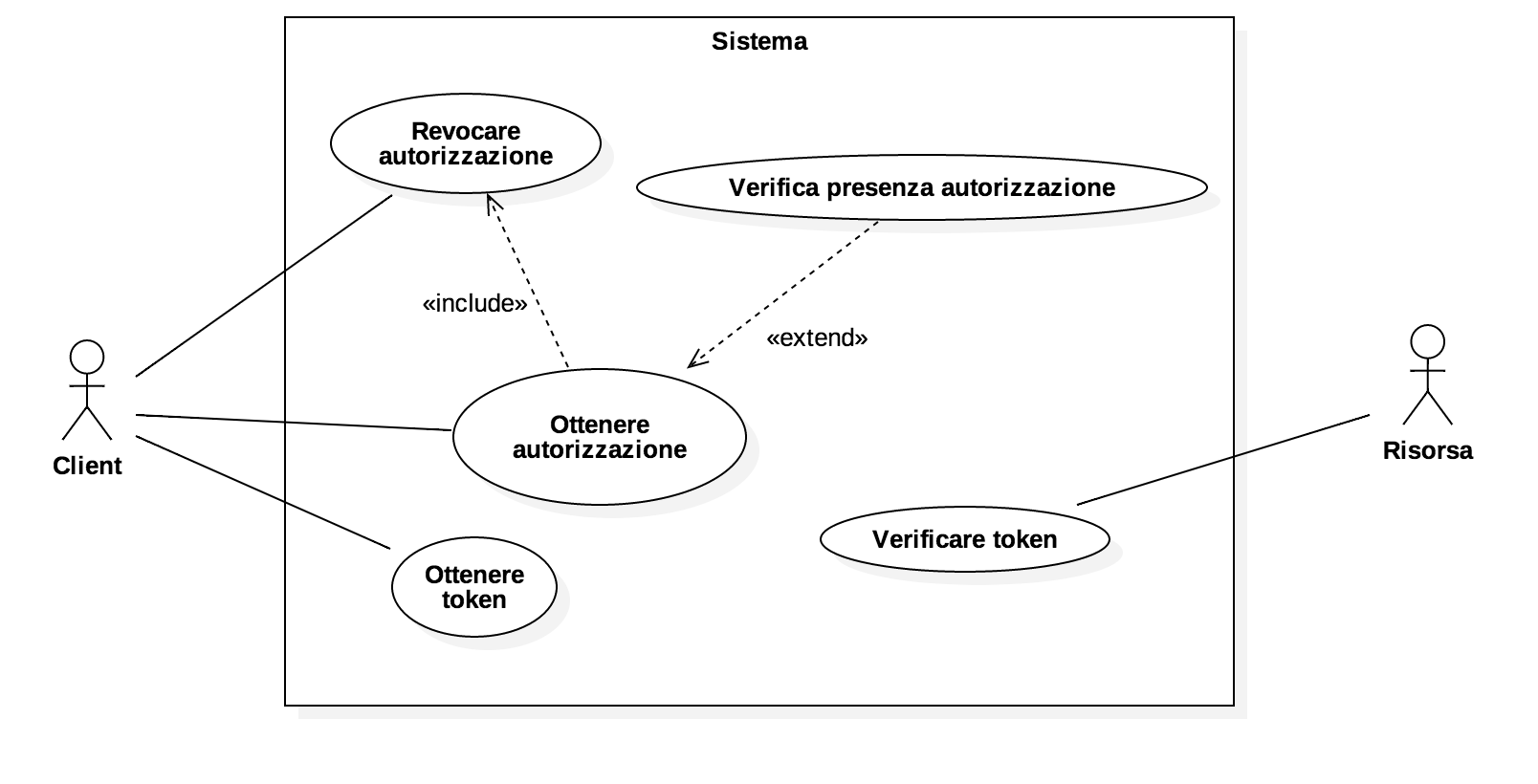
Lo studio del design UML per il progetto AuthOK assume un ruolo di raccordo tra l’analisi dei requisiti e la successiva fase di implementazione. Ciò viene eseguito attraverso la specifica e la documentazione del design dell’intero software. Per svolgere questo passaggio si ricorre all’uso di diversi 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.

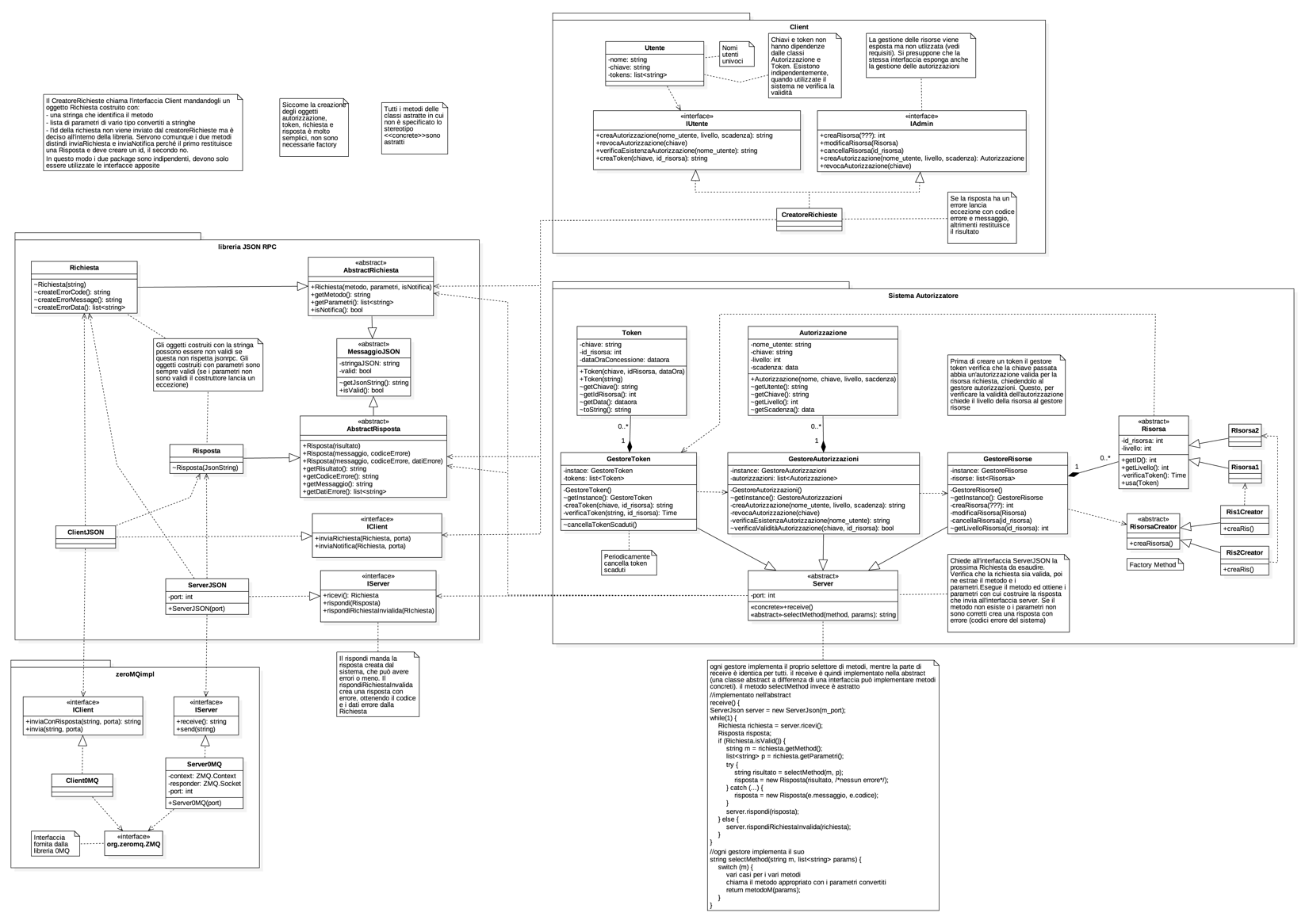
**UML - Unified Modeling Language**

## **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.

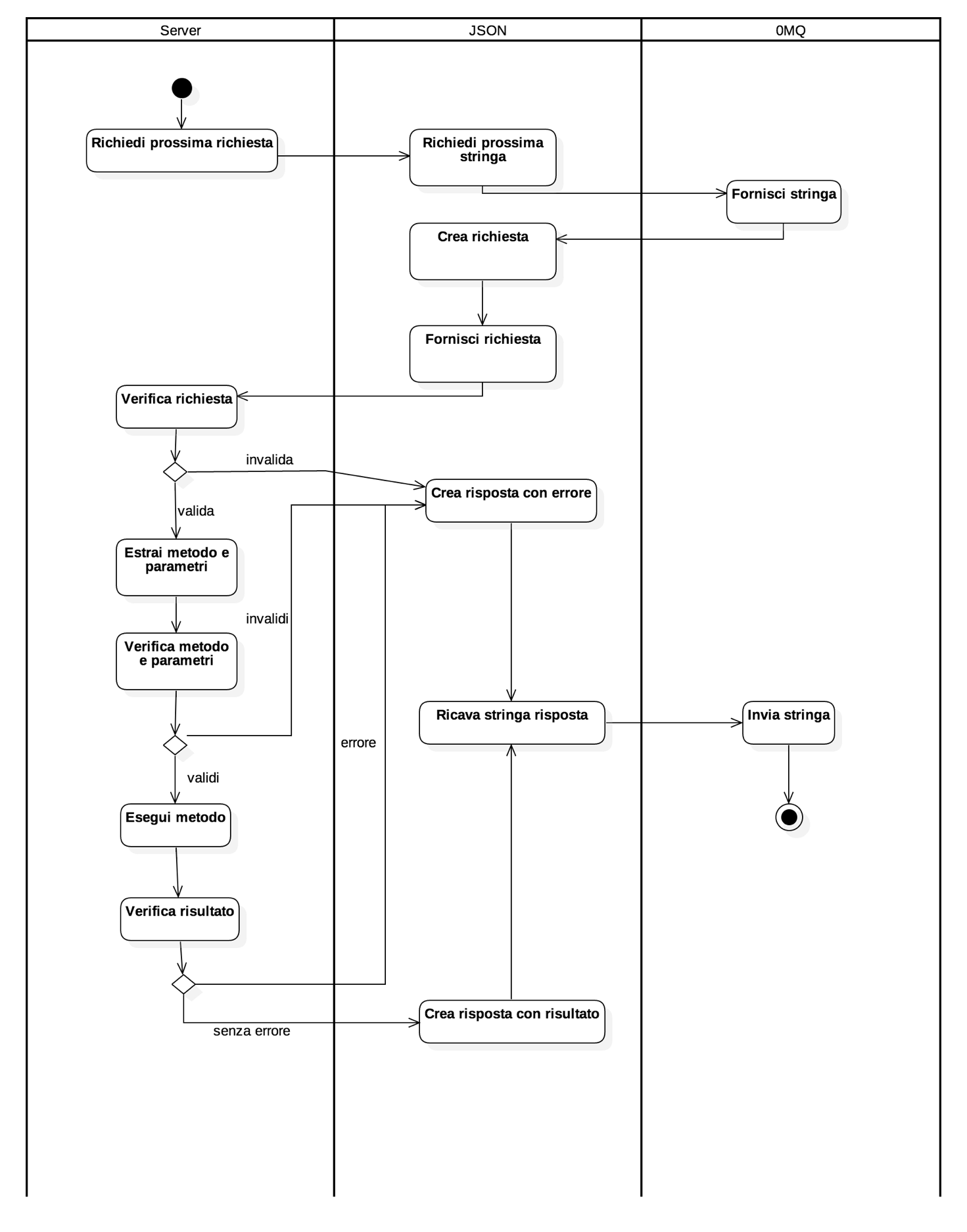
Una generica risorsa potrà interagire con il sistema per verificare la validità di un token in qualunque momento.

**Class Diagram**

[descrizione dettagliata]

## **Activity Diagram**

### **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 e 0MQ

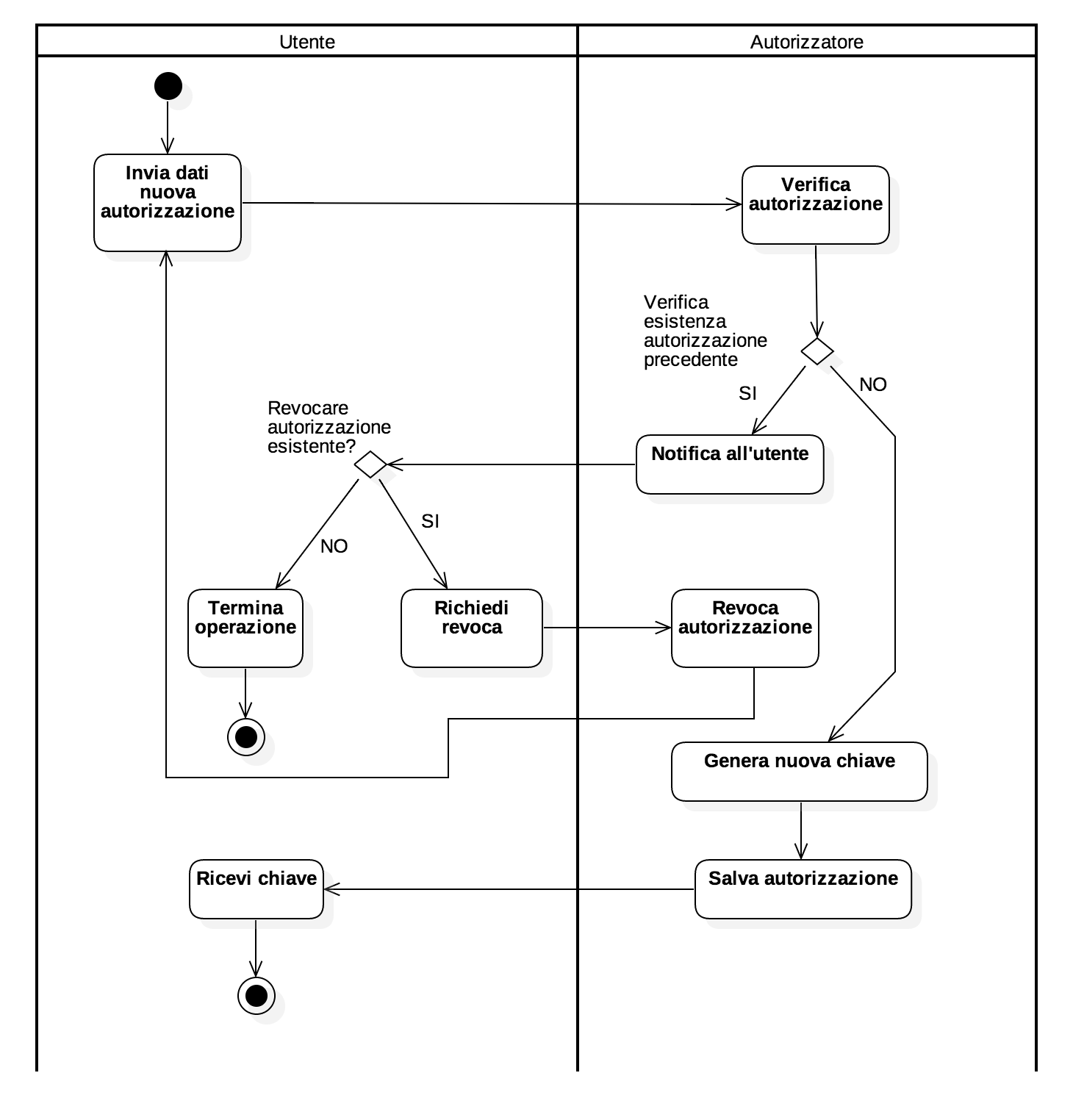
Si assume che il server rimanga in attesa di nuove richieste da parte del client. A sua volta anche il server JSON attenderà la disponibilità di una nuova stringa da elaborare che invierà alla libreria 0MQ. Tale libreria ha il ruolo di canale di comunicazione tra il server e il client JSON.

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 per poter procedere alla ritrasmissione all’utente.

### **Activity Diagram – Creazione autorizzazione**

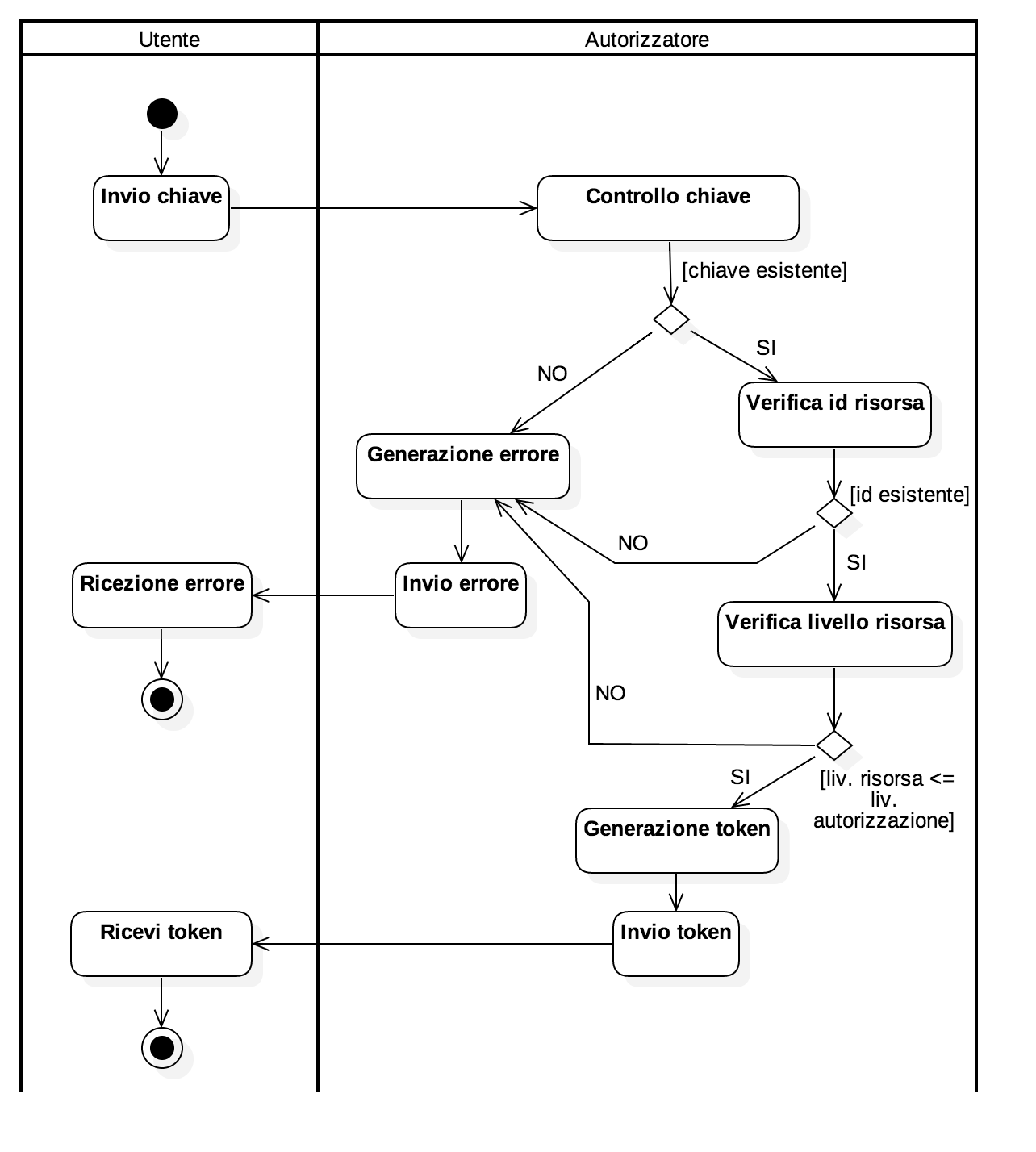


Si ritiene significativo mostrare con il seguente diagramma i passaggi eseguiti dall’utente e dall’autorizzatore nel processo di creazione di una nuova chiave.

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.  
  
**dopo revoca autorizzazione bisogna richiedere nuovamente la creazione o si può andare direttamente a genera chiave?**

### **Activity Diagram – Creazione token**

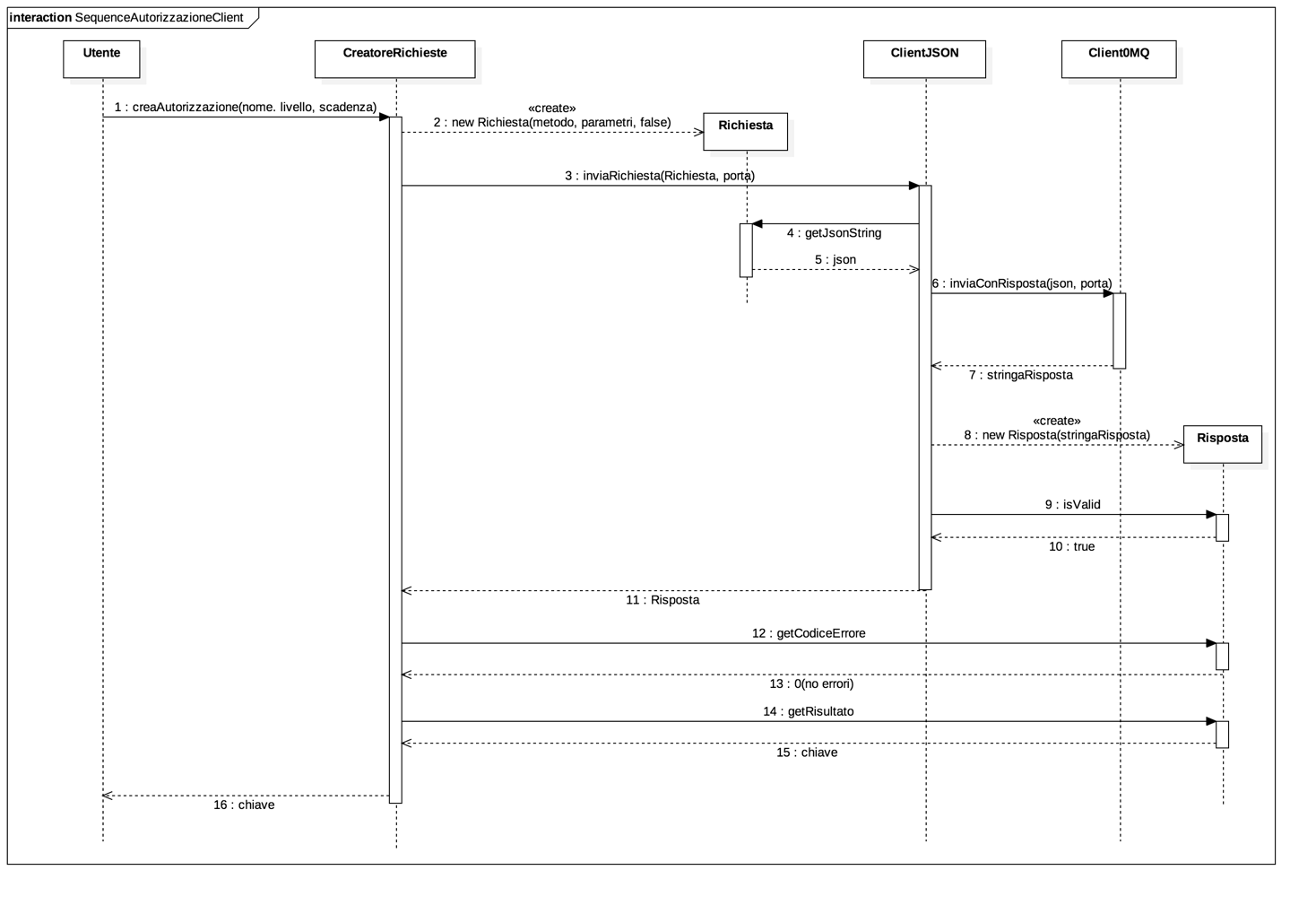
****

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 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 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 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.

### ../../SequenceAutorizzazioneServer.png**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, dal server 0MQ, dalla richiesta e dalla risposta. Il gestore autorizzazioni rimane in attesa di ricevere nuove richieste dal server JSON 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 “getJsonString” che ne consentirà la trasmissione all’utente come indicato nel primo Sequence Diagram riportato a pagina 9.

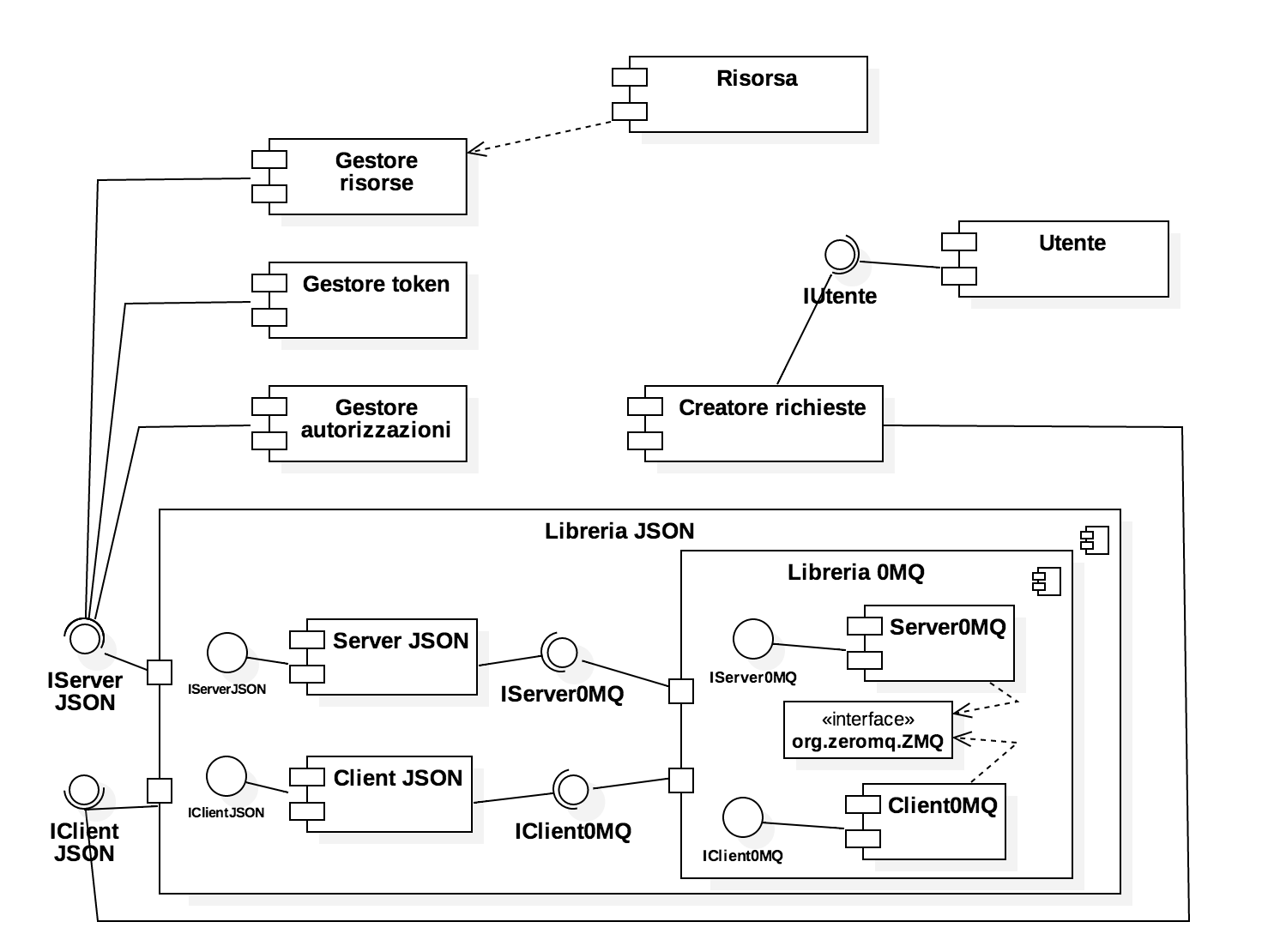
La chiamata 25 "rispondi" viene effettuata utilizzando come parametro l'oggetto Risposta che è stato creato in uno dei tre possibili rami. Lo stesso oggetto sarà quello invocato dalla chiamata 26 getJsonString e che restituisce la risposta 27 jsonString. Durante l'esecuzione uno solo dei rami verrà seguito e di conseguenza ci sarà una sola istanza di Risposta da inviare e quindi da cui ricavare la stringa json.

## **State Diagram**

### **State Diagram – Autorizzazione client**

**State Diagram –**

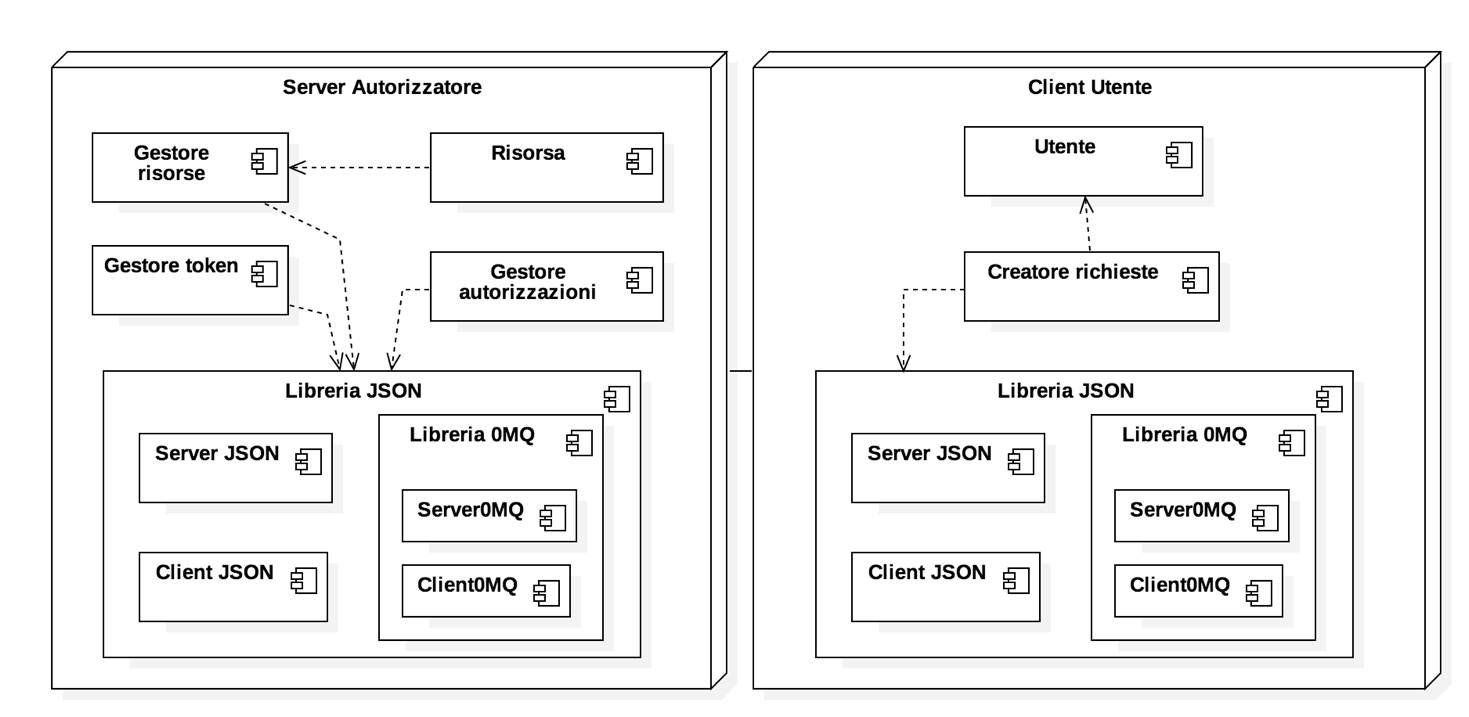
## **Component Diagram**



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 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.  
  
**bisogna aggiungere i component sistema e utente?**

## **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 viene inserito e sviluppato in entrambe i nodi in quanto svolge un ruolo centrale nella comunicazione delle due parti.

**verificare direzione freccie  
verificare che sia possibile ripetere la libreria json nei nodi**

## **Object Diagram**

## **Collaboration Diagram**

**Identificativo gruppo**: 10

**Progetto**: Tema A

**Componenti**:

845386 Ferrario Stefano [stefano6.ferrario@mail.polimi.it](mailto:stefano6.ferrario@mail.polimi.it)

843994 Gumus Tayfun [tayfun.gumus@mail.polimi.it](mailto:tayfun.gumus@mail.polimi.it)

854412 Isella Paolo [paolo.isella@mail.polimi.it](mailto:paolo.isella@mail.polimi.it)

845326 Martinese Federico [federico.martinese@mail.polimi.it](mailto:federico.martinese@mail.polimi.it)