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 I – Requisiti

Ferrario Stefano
Gumus Tayfun
Isella Paolo
Martinese Federico

Indice

Introduzione

Il nostro progetto è “AuthOK”, un sistema di autenticazione basato su un'architettura client server.

Il progetto prevede la realizzazione di una libreria in Java per la comunicazione remota tra sistemi. La libreria implementa in modo preciso la specifica del protocollo JSON-RPC e consente di implementare client e server JSON-RPC:

- Un server poterà ricevere richieste o notifiche e inviare risposte.
- Un client potrà inviare richieste o notifiche e ricevere risposte.

Sulla base di questa libreria viene realizzato un **sistema di autorizzazione centralizzato** per l’accesso a certe risorse. Il sistema centrale detiene l’elenco delle risorse accessibili. Tale elenco è aggiornabile creando, modificando o cancellando risorse. Analogamente il sistema tiene traccia delle autorizzazioni concesse. Deve essere possibile:

- Richiedere nuove autorizzazioni: fornendo i dati dell’utente, un livello e una scadenza, il sistema genera una chiave unica segreta per l’utente che dura fino alla scadenza. La chiave è associata ad un livello di autorizzazione: l’utente risulta autorizzato ad accedere a tutte le risorse associate a un livello inferiore o uguale alla propria autorizzazione.
- Revocare autorizzazioni (data la chiave)
- Chiedere un token di accesso a una risorsa: il sistema riceve la richiesta con il codice della risorsa e la chiave utente, verifica se la chiave è compatibile con il livello di risorsa richiesto e in caso positivo genera un token della durata di 24 ore per accedere alla risorsa.

Una risorsa può verificare sul sistema in ogni momento la validità di un token: dato token valido, il sistema restituisce per quanto tempo è ancora valido. Dato un token scaduto o inesistente, il sistema restituisce errore.

Assunzioni

Al fine di evitare l’introduzione di ulteriori complicazioni nel progetto e vista l’assenza di specifiche a riguardo, si assume possibile che *qualsiasi utente* possa autorizzare *chiunque*. Il sistema di autorizzazione è quindi non sicuro da specifica.
Un'altra assunzione riguarda la gestione delle risorse, in particolare riguardo creazione, modifica e cancellazione delle stesse. Non essendo specificato *quale attore* agisca con l'obiettivo di modificare l'elenco delle risorse (ad esempio creandone una nuova) si è deciso di *non* assegnare tale obiettivo al client. La gestione delle risorse rimane quindi un goal interno del sistema autorizzatore, che è in grado di gestire queste funzionalità, ma non è un *dependee* per alcun obiettivo esterno.

Ricerca della soluzione

Al fine di effettuare l’analisi dei requisiti siamo scesi nel dettaglio utilizzando gli strumenti che ci sono stati forniti. In parallelo alla redazione di un Data Dictionary, nel quale vengono descritte in modo più approfondito le entità principali del progetto, abbiamo sviluppato l’SDM e l’SRM del Goal Diagram. A causa della semplicità del diagramma I* riguardante il tema comune (libreria

JSON-RPC) abbiamo ritenuto superfluo mostrare sia l'SDM sia l'SRM.
All'interno dei diagrammi abbiamo scelto attori, obiettivi, relazioni strategiche e, dopo aver esteso i vari agenti e ruoli, ne abbiamo definito le attività principali.

Data Dictionary

Il processo di analisi dei requisiti ci ha portato alla realizzazione di due Data Dictionary. Il primo riguarda il tema comune del progetto, che prevede la realizzazione dell'interfaccia di comunicazione client-server mediante l'implementazione della libreria JSON.
Nel secondo Data Dictionary che presentiamo abbiamo analizzato gli elementi che a nostro giudizio costituiscono i concetti fondamentali per la realizzazione e il corretto funzionamento di un sistema di autenticazione
In entrambi i casi, per ciascun elemento vengono elencati una serie di attributi che ne permettono la descrizione in tutte le sfaccettature in modo da realizzare una visione più completa ed efficace di ogni entità.

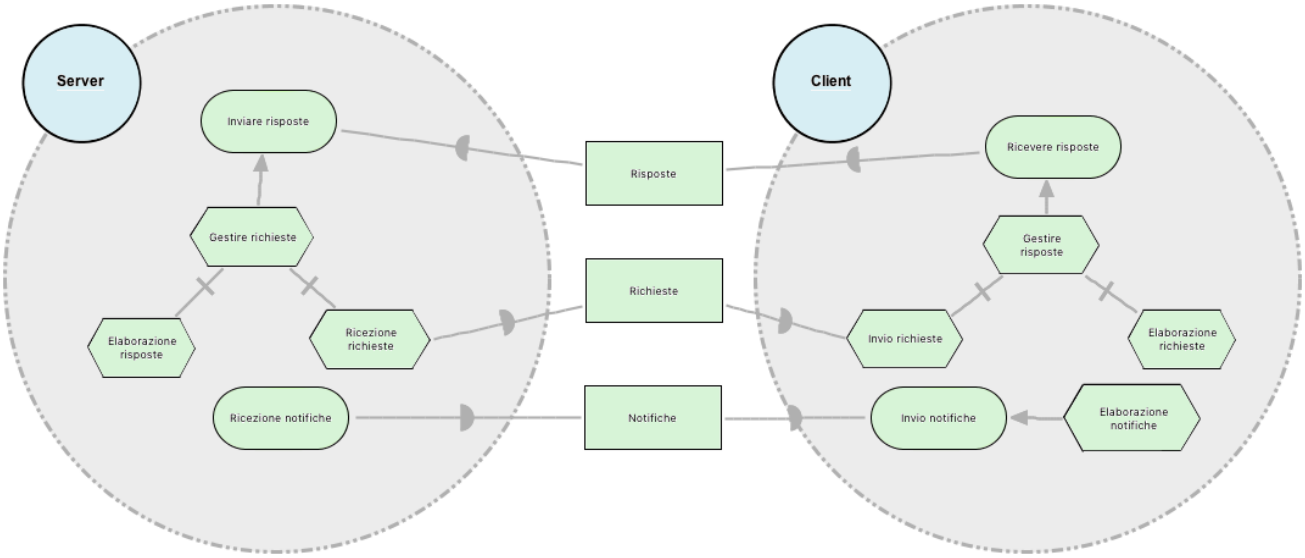
Tema comune					
Nome	**Sinonimi**	**Descrizione**	**Relazioni**	**Esempi**	**Attributi**
Richiesta		Chiamata rpc da parte di un client a un server. Specifica la versione di json rpc usata (2.0), il metodo da invocare, i parametri necessari e un identificatore della richiesta		{"jsonrpc": "2.0", "method": "subtract", "params": [42, 23], "id": 1} {"jsonrpc": "2.0", "method": "foobar", "id": "1"}	jsonrpc (sempre "2.0") method params id
Risposta		Risposta del server ad una richiesta (non notifica) del client. Specifica la versione di json rpc usata (2.0) e l'id della richiesta cui si riferisce. Contiene o il risultato del metodo invocato o un oggetto errore.	Errore	{"jsonrpc": "2.0", "result": 19, "id": 1} {"jsonrpc": "2.0", "error": {"code": -32601, "message": "Method not found"}, "id": "1"}	jsonrpc (sempre "2.0") result / error id
Notifica		È una richiesta senza un id. Indica che il client non vuole ricevere una risposta alla richiesta.		{"jsonrpc": "2.0", "method": "update", "params": [1,2,3,4,5]} {"jsonrpc": "2.0", "method": "foobar"}	jsonrpc (sempre "2.0") method params
Errore		Membro di una risposta quando il server riscontra un errore. Identifica il tipo di errore riscontrato tramite codice, un messaggio che descrive l'errore e dei dati aggiuntivi opzionali	Risposta	"error": {"code": -32601, "message": "Method not found"} "error": {"code": -32700, "message": "Parse error"}	code message data

Tema A					
Nome	**Sinonimi**	**Descrizione**	**Relazioni**	**Esempi**	**Attributi**
Risorsa	servizio	Servizio accessibile dagli utenti autorizzati. Identificata con id univoco, ha un livello di sicurezza.	Livello Codice risorsa Token	<1234, risorsa1, 7> <8743, risorsa2, 3>	ID risorsa Nome Livello
Autorizzazione		Associa un utente, identificato tramite la combinazione id utente – chiave segreta, ad un livello di autorizzazione e ad una scadenza. Fino alla data di scadenza un utente con la chiave specificata è autorizzato ad accedere a risorse con livello di sicurezza uguale o inferiore a quello indicato	Utente Chiave Livello Scadenza	<100, chiave1, 7, 31/12/2018> <28, chiave2, 5, 20/4/2019>	ID utente Chiave Livello Scadenza
Utente	dati utente Client	Identifica gli utenti che chiedono autorizzazioni per poter accedere alle risorse. Identificati da un id univoco	Autorizzazione	<100, Mario> <28, Luigi>	ID utente Nome
Token		Permesso rilasciato dal sistema ad un utente per accedere ad una determinata risorsa. È rilasciato solo dopo aver verificato che una determinata chiave sia associata ad un'autorizzazione valida. È valido per 24 ore a partire dal momento della concessione. La risorsa può chiedere al sistema di verificarne la validità.	Risorsa Chiave	<1234, 29/11/2018 14:00:00, chiave1> <8743, 24/02/2019 09:30:00, chiave2> <8743, 30/12/2018 12:00:00, chiave1>	ID risorsa Data e ora concessione Chiave
Livello		Identifica il livello di sicurezza di una risorsa ed il livello di permesso associato ad una autorizzazione. Un utente con livello n può accedere a tutte le risorse con livello <= n	Risorsa Autorizzazione	5 7 3	/
Scadenza		Specifica fino a che data una autorizzazione è valida	Autorizzazione	31/12/2018 20/4/2019	/
Chiave segreta	password	Stringa univoca per utente e segreta che il sistema crea ed assegna a ciascun utente. È associata ad una autorizzazione	Autorizzazione Token	chiave1 chiave2	/
ID risorsa	codice risorsa	Identificativo (numerico) univoco di una risorsa	Risorsa	1234 8743	/

Goal Diagram

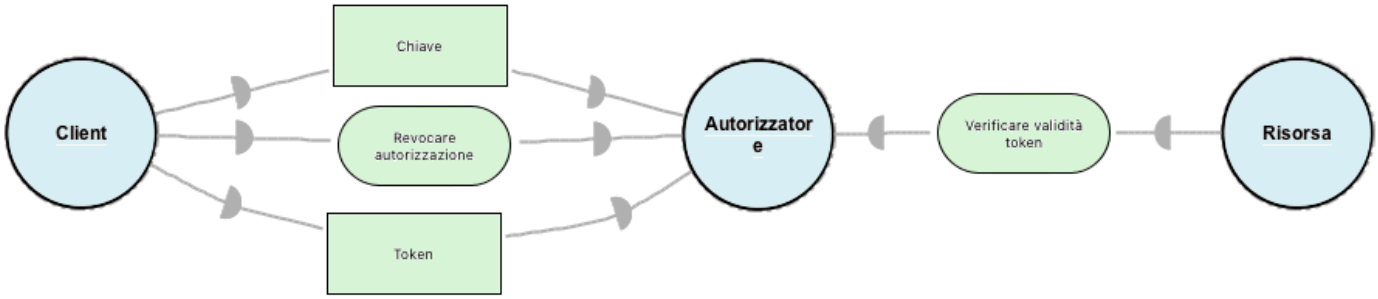
La stesura del Goal Diagram relativa al progetto ha come obiettivo primario quello di mantenere una struttura il più possibile semplice ed intuitiva in modo da fornire uno strumento per la comprensione immediata del funzionamento ad alto livello dell'applicazione mediante la scelta di attori, relazioni, obiettivi e risorse.

Tema Comune (libreria JSON-RPC)



Il diagramma ci guida a una prima visione del progetto. Partiamo con una rappresentazione grafica del funzionamento della libreria JSON che vogliamo implementare. Tale libreria gestisce il formato di comunicazione tra un client ed un server, attori del diagramma.
Come si può notare dal grafico, il server riceve richieste e notifiche dal client che a sua volta riceve risposte dal server. L'invio e la ricezione di richieste, risposte e notifiche (tutte identificate come risorse) sono gli obiettivi dei due attori e vengono raggiunti attraverso i task indicati.

Tema A - SDM (Strategic Dependency Model)



Abbiamo identificato tre attori; denominati **Client**, **Autorizzatore** e **Risorsa**; e due obiettivi principali, nello specifico:

- Per il client
  - **revocare autorizzazioni**
- Per la risorsa
  - **verificare la validità di un token**

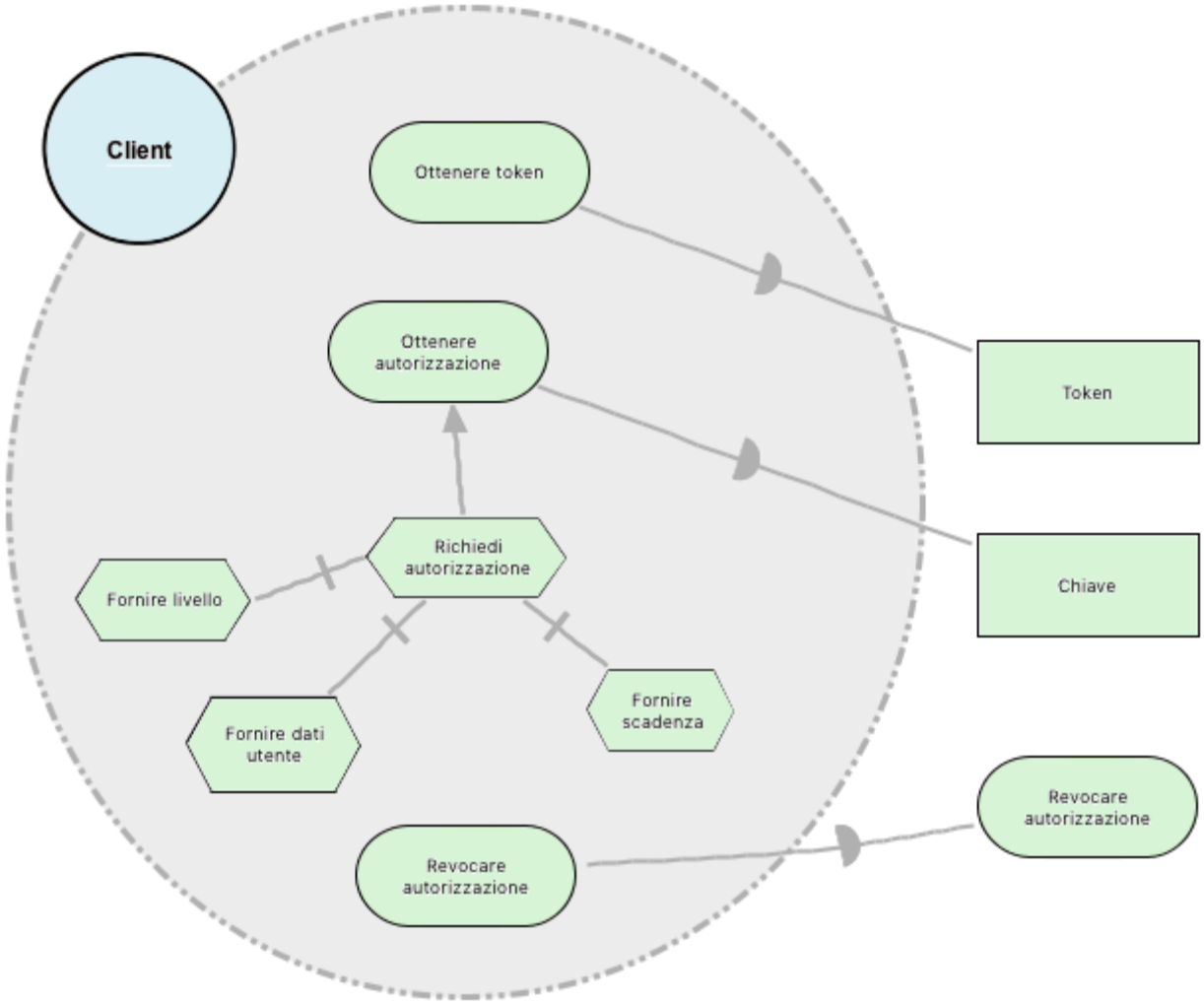
Nel grafico vengono poi rappresentate le risorse “Chiave” e “Token” che l’Autorizzatore invia al client il quale ha l’obiettivo di ottenere un’autorizzazione e di ottenere un token per l’accesso a una Risorsa.
Si noti che l'attore Client del tema comune è un elemento differente dall'attore Client del tema A. Il primo è una delle due entità che comunicano attraverso un protocollo implementato dalla libreria, mentre il secondo è l'utilizzatore di un servizio di autorizzazione e di accesso a risorse.

Per raggiungere i suoi obiettivi il Client dipende dall’Autorizzatore. La Risorsa, invece, dipende dall'Autorizzatore per poter verificare la validità di un token.

Il sistema non prevede l’implementazione dell’accesso da parte di un Client ad una Risorsa.

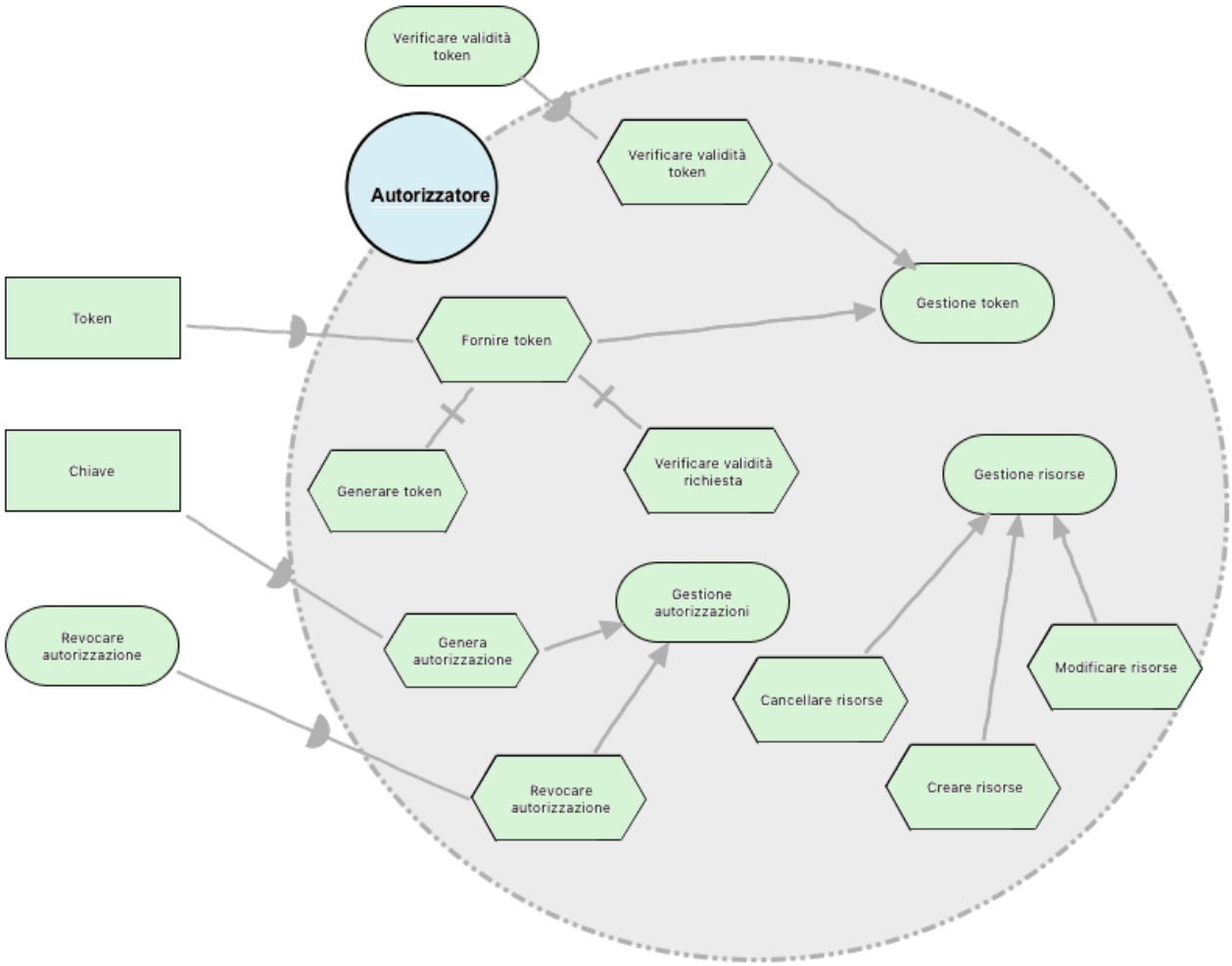
Dopo aver completato l'analisi dei goal e delle dipendenze strategiche si è passati alla realizzazione di uno schema **SRM (Strategic Rationale Model)**, che riportiamo scomposto nei diversi attori per comodità di lettura.

SRM – Client



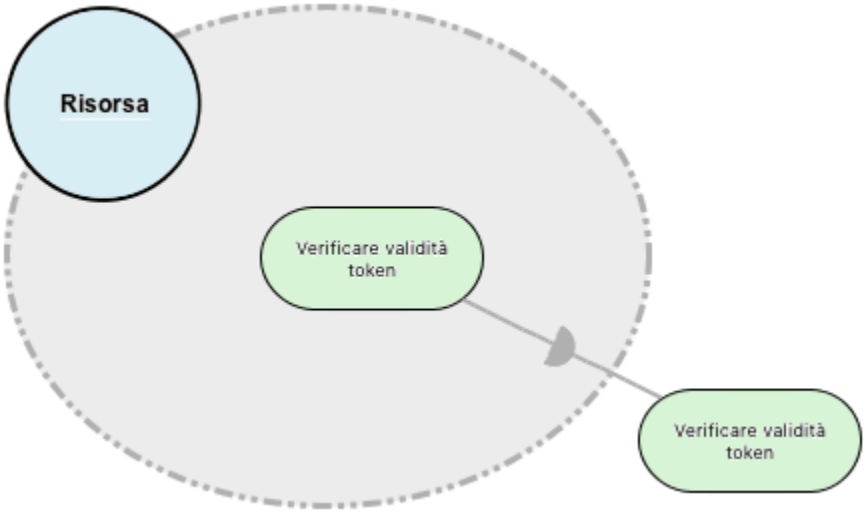
Il primo goal da soddisfare per il Client è **ottenere un'autorizzazione**. Per fare ciò si dovrà procedere con l'esecuzione dei task che portano alla compilazione di una richiesta, fornendo le informazioni necessarie. Il Client ha inoltre la possibilità di **revocare un'autorizzazione**, identificato come ulteriore obiettivo. Poiché nelle specifiche non viene precisato tale aspetto, ammettiamo che un utente qualsiasi possa richiedere e revocare l'autorizzazione per chiunque. Per accedere ad una risorsa sarà necessario **ottenere un token**, goal raggiungibile attraverso la richiesta di assegnamento di una chiave segreta.

SRM – Autorizzatore



L'Autorizzatore **gestisce le risorse** mediante le attività di creazione, modifica e cancellazione. In modo analogo **gestisce le autorizzazioni**, consentendo quindi la creazione di una nuova autorizzazione e la revoca. La concessione di un'autorizzazione prevede la **generazione di una chiave segreta** che viene restituita al Client come risorsa. L'Autorizzatore inoltre si occupa della **generazione di un token**. Per fare ciò procede alla verifica della richiesta proveniente dal Client ed una volta generato il token lo restituisce al Client come risorsa. L'Autorizzatore inoltre permette alla Risorsa di soddisfare l'obiettivo di **verifica la validità di un token** eseguendo il task necessario.

SRM – Risorsa



La Risorsa ha la possibilità di verificare in qualunque momento la validità di un token che le viene fornito. Nel seguente progetto non viene implementato l'accesso da parte del Client ad una Risorsa.