Politecnico di Milano
Polo Territoriale di Como


# Prova finale di Ingegneria del Software
# Documento di progetto

- Gruppo 1 -

Ghielmetti Nicolò
Quaglia Ennio
Zaffaroni Leonardo

# Indice

# Tema del Progetto

## Tema Comune

Realizzare una libreria in Java per la comunicazione remota tra sistemi.
La libreria deve implementare **in modo preciso** la specifica del protocollo JSON-RPC.
I dettagli del protocollo sono descritti qui: http://www.jsonrpc.org/specification
La libreria deve consentire di implementare client e server JSON-RPC:

- Un server deve poter ricevere richieste o notifiche e inviare risposte (ove queste siano previste dalla specifica).
- Un client deve poter inviare richieste o notifiche e ricevere risposte (ove queste siano previste dalla specifica).

Allo scopo di garantire la comunicazione remota, è necessario selezionare un "canale di comunicazione" su cui il protocollo è implementato.
Si suggerisce di rendere la libreria agnostica rispetto al canale di comunicazione, tramite un'apposita interfaccia e/o factory.
Come canale si suggerisce fortemente l'uso di un approccio a code, come ad esempio ZeroMQ.
Le risorse di ZeroMQ sono accessibili qui: http://zeromq.org/ (e verranno introdotte a lezione).

Una volta realizzato il tema comune, è possibile utilizzare la libreria ottenuta per implementare una specifica applicazione. In particolare:

- Ogni gruppo può scegliere se realizzare l'applicazione A) o B)
- Inoltre, è possibile ottenere 3 punti di bonus aggiuntivo se due gruppi si coordinano per usare le due applicazioni insieme: vedere descrizione dell'applicazione I (Integration).

## Tema B) Broker

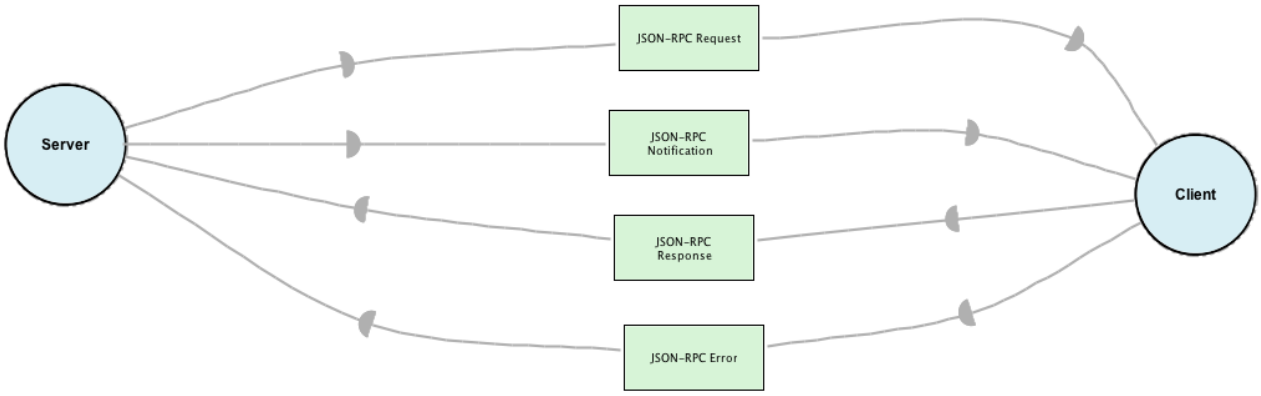
Si vuole realizzare un sistema che fa da broker per servizi online, in modo trasparente.
Il sistema detiene una lista di servizi online disponibili e consente di registrare nuovi servizi o cancellarne di esistenti.
Ogni servizio è descritto da un identificativo di servizio e un formato di input e output. Inoltre, sono disponibili una serie di metadati: proprietario, settore, parole chiave, titolo, descrizione, data di attivazione.
Il broker associa ad ogni servizio un metodo JSON-RPC che lo identifica in modo univoco e che ne permette l'invocazione trasparente (più client possono esporre servizi con lo stesso identificativo di servizio, ad esempio "Somma").
E' possibile richiedere al broker (anch'esso implementato tramite metodi JSON-RPC):

- Di restituire un elenco di servizi in base a una ricerca (una o più parola, che viene cercata in tutti i metadati). Il risultato è una lista di servizi con tutta la loro descrizione e il metodo JSON-RPC associato. Ad esempio se cerco "Somma" e ci sono più servizi con questo identificativo di servizio, il broker ritorna tutti questi servizi e il loro identificativo JSON-RPC ("SommaXX")
- Di invocare, in modo trasparente, un servizio, tramite il metodo JSON-RPC associato ("SommaXX").

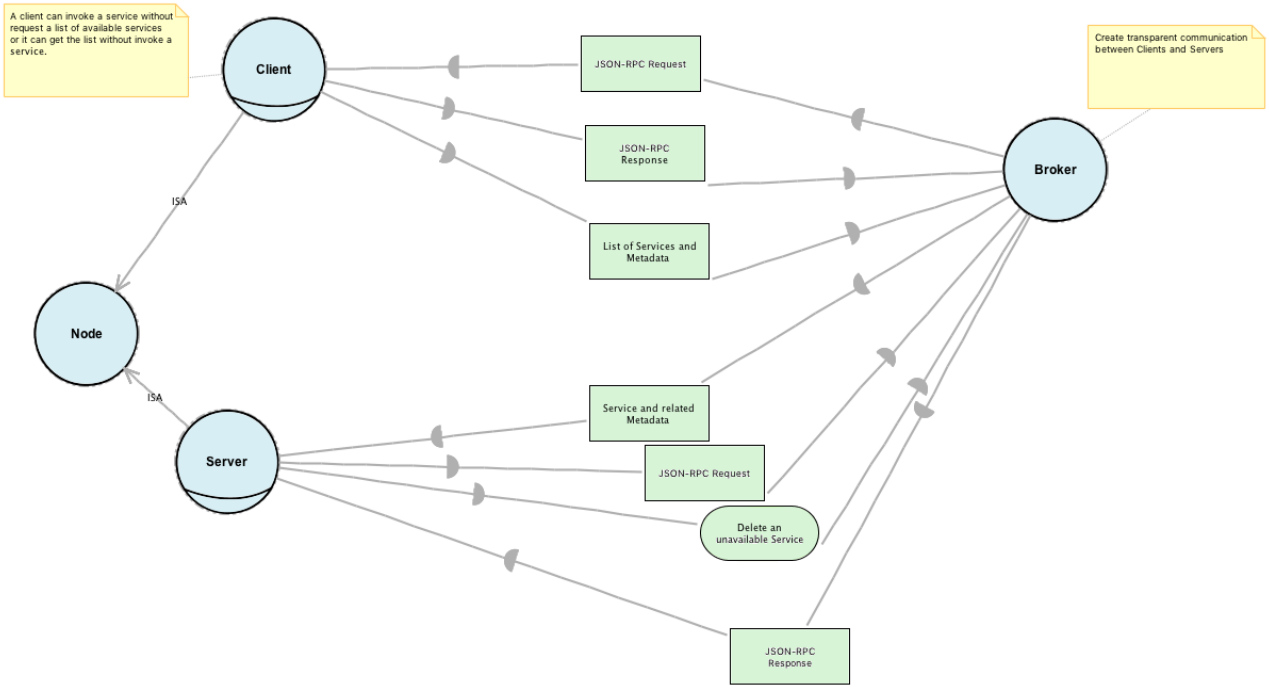
Quando un client si disconnette dal broker i suoi servizi vengono automaticamente cancellati ed eventuali richieste pendenti completate con il codice di errore più opportuno.
Si può supporre che il broker e i servizi siano invocabili tramite quanto specificato nel tema comune.

# Digrammi i* - SDM

## Tema Comune

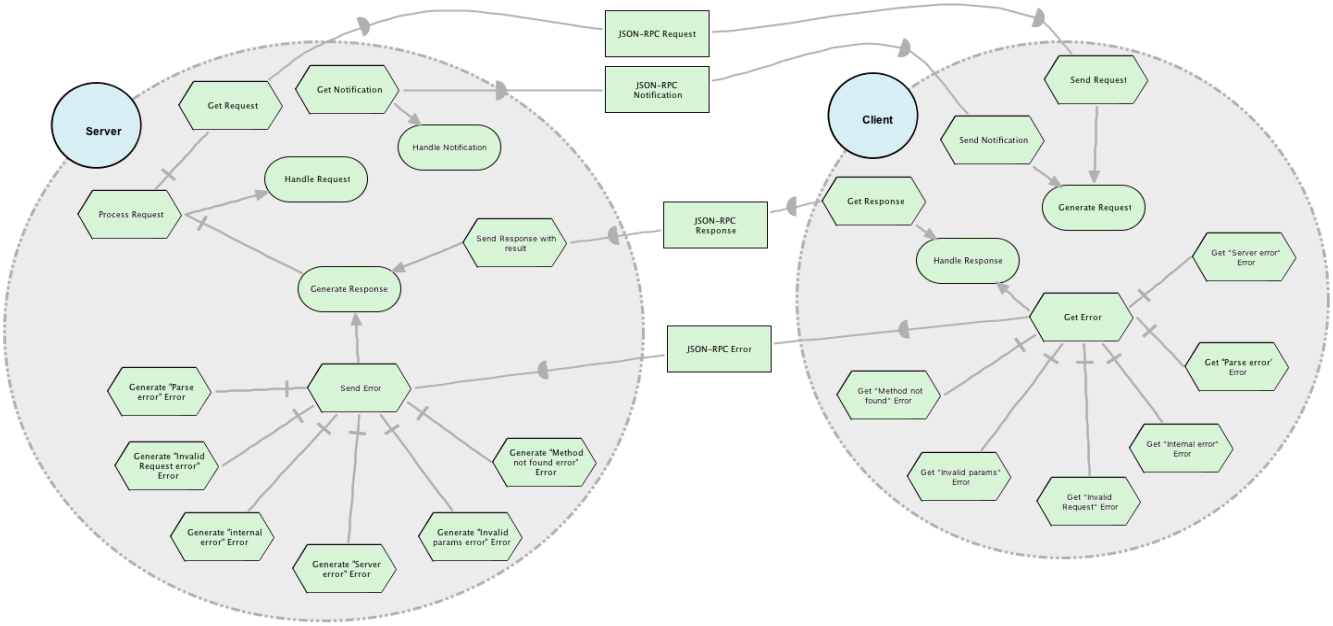



## Tema B

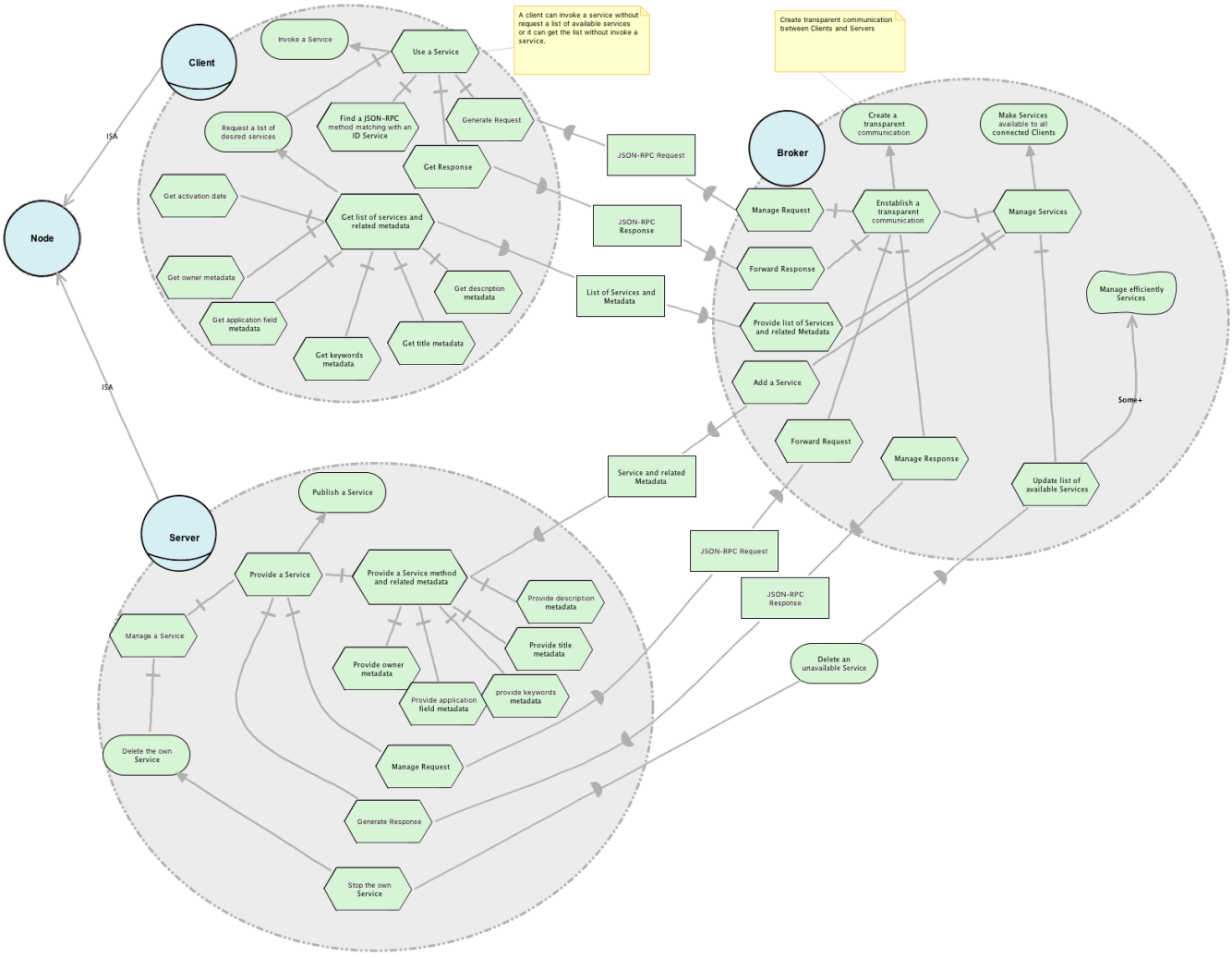


# Diagrammi i* - SRM

## Tema Comune




## Tema B



# Sintesi dei Goal

## Goal tema comune:

- Obiettivi Client
  - Generate Request:
    Al fine di comunicare con il Server è previsto dal protocollo JSON-RPC che il Client generi un oggetto JSON-RPC Request. Affinchè questo avvenga il Client dev'essere in grado di comporre, secondo lo standard, la richiesta da scambiare con il Server.
  - Handle Response:
    Per completare la comunicazione fra Client e Server è necessario che il Client sia in grado di accettare, gestendo eventuali errori, le JSON-RPC Response che riceverà dal Server in seguito ad una Request effettuata.
- Obiettivi Server
  - Generate Response:
    Per fornire al Client il risultato dell'operazione che ha richiesto oppure per segnalare un errore nella JSON-RPC Request, il Server deve comporre e inoltrare al Client, rispettivamente, un oggetto JSON-RPC Response e un oggetto JSON-RPC Error.
  - Handle Request:
    È necessario che il Server sia in grado di ricevere e interpretare secondo lo standard JSON-RPC le Requests che riceverà.

## Goal tema B) Broker

- Obiettivi Client
  - Invoke a Service:
    Invocare un servizio fra quelli registrati nel sistema (se si conosce il servizio non è necessario richiedere informazioni riguardo i servizi disponibili). Il Client deve poter dare la possibilità all'utente di usufruire dei servizi disponibili nel Broker.
  - Request a list of desired Services:
    Richiedere una lista, basata su una ricerca, di servizi registrati nel sistema. Al fine di comunicare all'utente come utilizzare ciascun servizio è necessario che il Client sia in possesso della lista.
- Obiettivi Server
  - Publish a Service:
    Pubblicare il servizio che il Server vuole fornire al sistema. Il Server deve poter essere raggiunto dal Broker qualora un Client volesse usufruire (attraverso il canale di comunicazione trasparente stabilito dal Broker) del suo servizio.
  - Delete the own Service:
    Dev'essere sempre possibile per il Server rimuovere, attraverso un'opportuna comunicazione, il proprio servizio dall'elenco dei servizi del Broker.

- Obiettivi Broker
  - Make Services available to all connected Clients:
    Per rendere disponibili i servizi registrati nel sistema, il Broker ha il compito di tenere aggiornato l'elenco dei servizi visibili ai Clients. In particolare l'operazione di aggiornamento consiste nell'aggiunta di nuovi servizi e rimozione di quelli non più disponibili.
  - Create a transparent communication:
    È compito del Broker fare da tramite fra il Client e il Server per l'intera durata della comunicazione, senza che il Client sia in possesso delle informazioni necessarie per dialogare con il Server.
    Le Request del Client vengono inoltrate (senza subire alcuna modifica) al Server il quale, allo stesso modo, attraverso il Broker invierà una Response al Client.
  - Manage efficiently services (Soft-Goal):
    Affinchè il sistema possa usufruire dei servizi disponibili in modo efficiente è buona norma progettare il Broker in modo che la lista dei servizi disponibili non contenga servizi duplicati o servizi non più disponibili.

- Obiettivi comuni (Server-Broker)
  - Delete an unavailable Service:
    Il Broker e il Server possono negoziare la cancellazione di un servizio: il Server può richiedere al Broker la cancellazione del suo servizio dall'elenco dei servizi e il Broker deve cancellare i servizi non più disponibili o raggiungibili nel sistema.

# Data Dictionary

**Premessa:** i termini "String", "Numbers", "Structured", "Array", "Primitive", "`null`" sono da intendersi come definiti nella specifica JSON (http://www.json.org). I simboli "`-->`" e "`<--`" sono da intendersi, rispettivamente, come messaggio emesso dal *Client* (e ricevuto dal *Server*) e messaggio emesso dal *Server* (e ricevuto dal *Client*). I termini "*Server*" e "*Client*" sono da intendersi contestualizzati nella parte di progetto (Applicazione o Libreria) di cui si sta facendo riferimento.

## Available Services

| | |
|---|---|
| **Definizione** | *Services* messi a disposizione dal *Broker* a tutti i *Clients* del sistema. |
| **Tipo di dato** | Lista di istanze di *Service* |
| **Derivato da** | *Service* |

## Broker

| | |
|---|---|
| **Definizione** | Un componente del sistema che si occupa di fornire ai *Clients* i servizi disponibili all'interno del sistema. |

## Client – Applicazione[1]

| | |
|---|---|
| **Definizione** | Un *Client* è un componente del sistema che attraverso il *Broker* effettua richieste ai *Server* che forniscono il servizio richiesto. |
| **Derivato da** | *Node* |

## Client - Libreria[2]

| | |
|---|---|
| **Definizione** | Un programma finalizzato ad inviare *JSON-RPC Requests* e ottenere *JSON-RPC Responses* attraverso lo standard JSON-RPC. |

[1] S'intende il significato che assume il termine nel contesto della parte di progetto relativo all'applicazione B.
[2] S'intende il significato che assume il termine nel contesto della parte di progetto relativo alla libreria JSON-RPC.

## Error

<table>
<tr><td>Definizione</td><td>Messaggio inviato dal Server al Client quando una richiesta non viene eseguita correttamente. Per i codici di errore con i messaggi corrispondenti fare riferimento alla sezione “Tipo di dato” di questa voce.</td></tr>
<tr><td>Tipo di dato</td><td>Structured:
<ul>
<li>code: valore numerico che specifica il tipo di errore. Questo campo deve essere specificato per ogni tipo di errore.
<table>
<tr><th>Codice (code)</th><th>Messaggio (message)</th><th>Significato</th></tr>
<tr><td>-32700</td><td>Parse error</td><td>Errore di traduzione dell’oggetto JSON ricevuto dal Server.</td></tr>
<tr><td>-32600</td><td>Invalid request</td><td>JSON-RPC Request non valida.</td></tr>
<tr><td>-32601</td><td>Method not found</td><td>Il metodo JSON-RPC non esiste o non è disponibile.</td></tr>
<tr><td>-32602</td><td>Invalid params</td><td>Parametri/o non validi/o per il metodo richiesto.</td></tr>
<tr><td>-32603</td><td>Internal error</td><td>Errore interno JSON-RPC.</td></tr>
<tr><td>Da -32000 a -32099</td><td>Server error</td><td>Riservati per errori definiti al di fuori della libreria.</td></tr>
</table>
</li>
<li>message: String che descrive brevemente l’errore riscontrato.</li>
<li>data: un valore di tipo Primitive o Structured contenente informazioni aggiuntive riguardanti l’errore riscontrato. Questo campo può essere omesso.</li>
</ul>
</td></tr>
<tr><td>Esempi di utilizzo</td><td><pre>--> {"jsonrpc": "2.0", "method": "foobar, "params": "bar", "baz]
<-- {"jsonrpc": "2.0", "error": {"code": -32700, "message": "Parse error"}, "id": null}</pre></td></tr>
<tr><td>Derivato da</td><td>JSON-RPC Response</td></tr>
</table>

## JSON-RPC Request

<table>
<tr><td>Definizione</td><td>Una risorsa che rappresenta una chiamata a procedura remota (rpc-call), cioè una richiesta inviata da un Client ad un Server al fine di usare un Service.</td></tr>
<tr><td>Tipo di dato</td><td>Structured:
<ul>
<li>jsonrpc: String rappresentante la versione di json-rpc.</li>
<li>method: String contenente il nome del metodo che deve essere invocato.</li>
<li>params: valore Structured rappresentante l’insieme dei parametri utilizzati nell’invocazione.</li>
<li>id: valore String o Numer stabilito dal Client per identificare la richiesta.</li>
</ul>
</td></tr>
<tr><td>Esempi di utilizzo</td><td><pre>{"jsonrpc": "2.0", "method": "sum", "params": [12, 6], "id": 1}</pre></td></tr>
</table>

## JSON-RPC Response

| | |
|---|---|
| **Definizione** | Una risorsa che rappresenta la risposta conseguita dal Server a seguito di una JSON-RPC Request ricevuta. |
| **Tipo di dato** | Structured:<br>• **jsonrpc:** String rappresentante la versione di json-rpc.<br>• **result:** attributo specificato solo in caso di successo dell'operazione. Il valore di questo attributo è determinato dal metodo invocato nel *Server*.<br>• **error**: attributo presente solo se si verifica un errore durante l'invocazione del metodo. Se presente deve contenere un oggetto di tipo *JSON-RPC Error*.<br>• **id:** valore String o Numer identico all'id della *JSON-RPC Request* associata. Se si verifica un errore durante l'identificazione dell'id relativo alla *JSON-RPC Request* associata questo campo deve essere nullo (`null`). |
| **Esempi di utilizzo** | `{"jsonrpc": "2.0", "result": 18, "id": 1}` |

## Notification

| | |
|---|---|
| **Definizione** | Un messaggio inviato dal *Client* al *Server* che ha lo scopo di fornire informazioni senza aspettarsi un messaggio di risposta. |
| **Tipo di dato** | *JSON-RPC Request* con l'id non specificato. |
| **Esempi di utilizzo** | `--> {"jsonrpc": "2.0", "method": "update", "params": [1,2,3,4,5]}`<br>`--> {"jsonrpc": "2.0", "method": "foobar"}` |
| **Derivato da** | *JSON-RPC Request* |

## Metadata

| | |
|---|---|
| **Definizione** | L'insieme delle informazioni, che descrivono un *Service*, scambiate tra gli attori del sistema. |
| **Tipo di dato** | Structured:<br>• **title**: String rappresentante il nome del *Service* associato.<br>• **description**: String rappresentante una breve descrizione del *Service* associato.<br>• **keywords**: Array of String rappresentante le parole chiave del *Service* associato.<br>• **applicationField**: String rappresentante il campo di applicazione del *Service* associato.<br>• **owner**: String rappresentante l'identificativo del fornitore del *Service* associato. |

## Method

| | |
|---|---|
| **Definizione** | Un identificativo utilizzato per invocare una procedura remota (RPC) disponibile nel sistema. |
| **Tipo di dato** | String |
| **Esempi di utilizzo** | `{"jsonrpc": "2.0", "method": "subtract", "params": [42, 23], "id": 1}`<br>"subtract" è il nome del metodo che si intende invocare attraverso la soprastante *JSON-RPC Request*. |

## Node

| | |
|---|---|
| **Definizione** | Un componente del sistema che può utilizzare e/o fornire servizi. Ovvero, un *Node* potrebbe assumere la funzione di *Client* e/o di *Server*. |

### Publish (a *Service*)

| | |
|---|---|
| **Definizione** | L'azione effettuata dal *Server* finalizzata a rendere disponibile il proprio *Service* all'interno del sistema. |

### Server – Applicazione[1]

| | |
|---|---|
| **Definizione** | Un *Server* è un componente del sistema che attraverso il *Broker* pubblica e fornisce ai *Client* rispettivamente il servizio erogato e la risposta associata a una richiesta ricevuta da un *Client*. |
| **Derivato da** | *Node* |

### Server – Libreria[2]

| | |
|---|---|
| **Definizione** | Un programma finalizzato a ricevere dai *Clients JSON-RPC Requests* ed elaborare *JSON-RPC Responses* da inviare in risposta alle *JSON-RPC Requests* associate. |

### Service

| | |
|---|---|
| **Definizione** | Una risorsa scambiata nel sistema rappresentante una funzionalità fornita da un *Server*. |
| **Tipo di dato** | Structured<br>• **method:** String rappresentante il nome (univoco) del metodo associato alla risorsa.<br>• ***Metadata:*** metadata relativi alla risorsa. Vedasi voce "*Metadata*" per i dettagli sulla composizione di questo attributo. |

### Transparent communication

| | |
|---|---|
| **Definizione** | Il *Broker* permette ai *Clients* di invocare *Services* prendendosi carico della gestione della comunicazione tra i *Nodes*. *JSON-RPC Responses* e *JSON-RPC Requests* vengono inoltrate al *Broker* il quale si occupa di consegnarle ai rispettivi destinatari senza che il *Client* conosca direttamente il *Server* e viceversa. |