Politecnico di Milano

Polo Territoriale di Como



Prova finale di Ingegneria del Software  
Documento di progetto – Diagrammi UML

- Gruppo 1 -

Ghielmetti Nicolò

Quaglia Ennio

Zaffaroni Leonardo

Indice

[UseCase Diagram 3](#_Toc498960422)

[Component Diagram 4](#_Toc498960423)

[Deployment Diagram 5](#_Toc498960424)

[Class Diagram 6](#_Toc498960425)

[Object Diagram 7](#_Toc498960426)

[Communication Diagram 8](#_Toc498960427)

[Activity Diagram – Cancellazione di un servizio 9](#_Toc498960428)

[Activity Diagram – Invocazione di un servizio 10](#_Toc498960429)

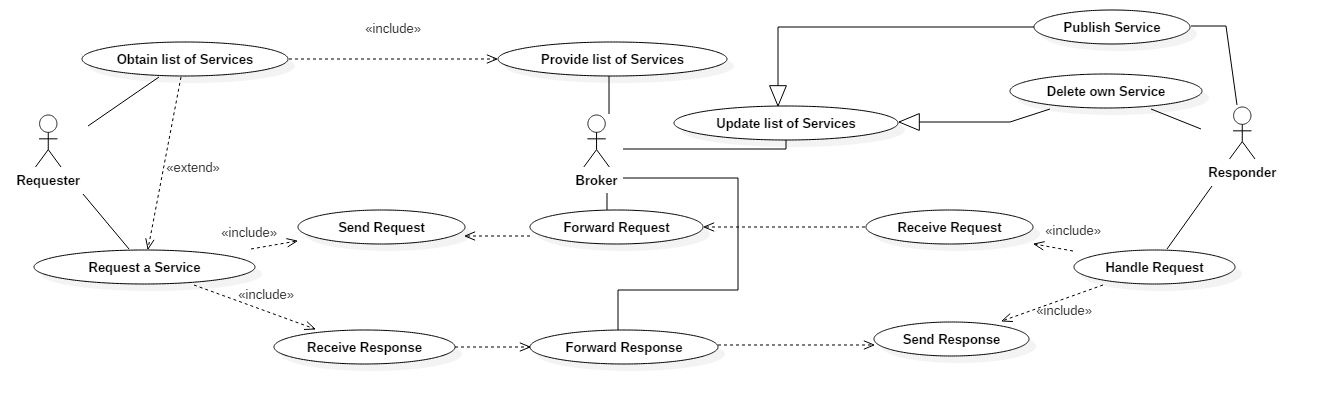
[Sequence Diagram – Richiesta della lista dei servizi 11](#_Toc498960430)

[Sequence Diagram – Invocazione di un servizio 12](#_Toc498960431)

[State Diagram – Broker 13](#_Toc498960432)

[State Diagram – Node 14](#_Toc498960433)

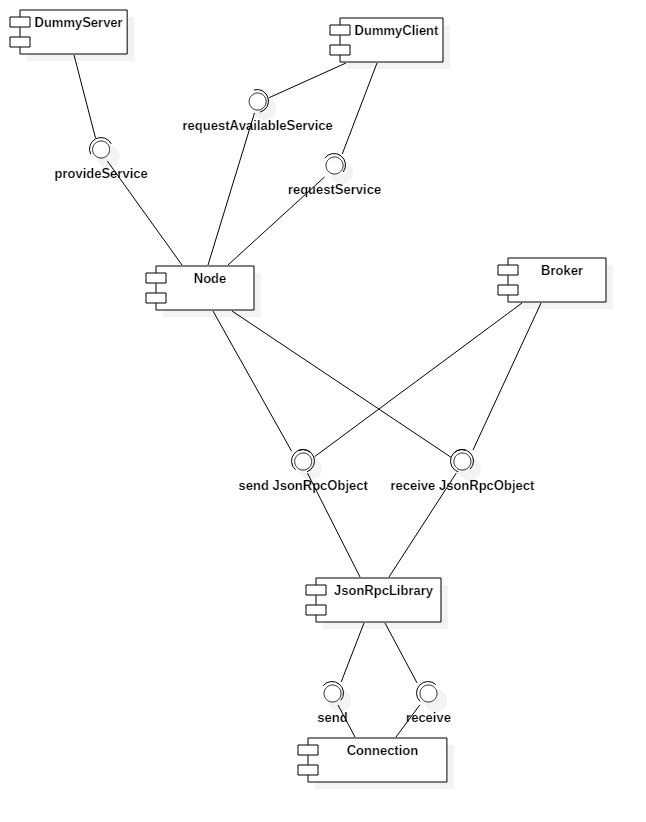
# UseCase Diagram



Il sistema si può rappresentare attraverso tre attori:

* Requester: è un utilizzatore di servizi. In particolare può:
  + richiedere una lista dei servizi disponibili al Broker.
  + richiedere l’invocazione di un servizio mediante la comunicazione trasparente realizzata dal Broker.
* Responder: è un fornitore di servizi. Ha il compito di:
  + gestire opportunamente le richieste, inviando risposte o errori adeguati.
  + gestire la sua presenza all’interno del sistema, aggiungendo e/o cancellando il proprio servizio.
* Broker: si occupa di gestire la comunicazione trasparente fra i vari Requesters e Responders presenti all’interno del sistema. In particolare si occupa di:
  + inoltrare richieste e risposte agli opportuni destinatari.
  + gestire la lista dei servizi disponibili mantenendola aggiornata.

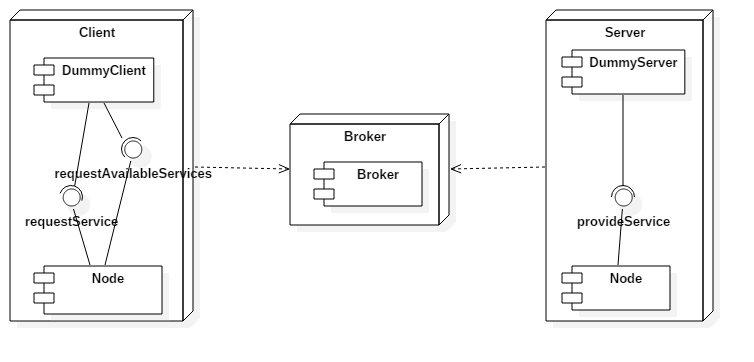
# Component Diagram



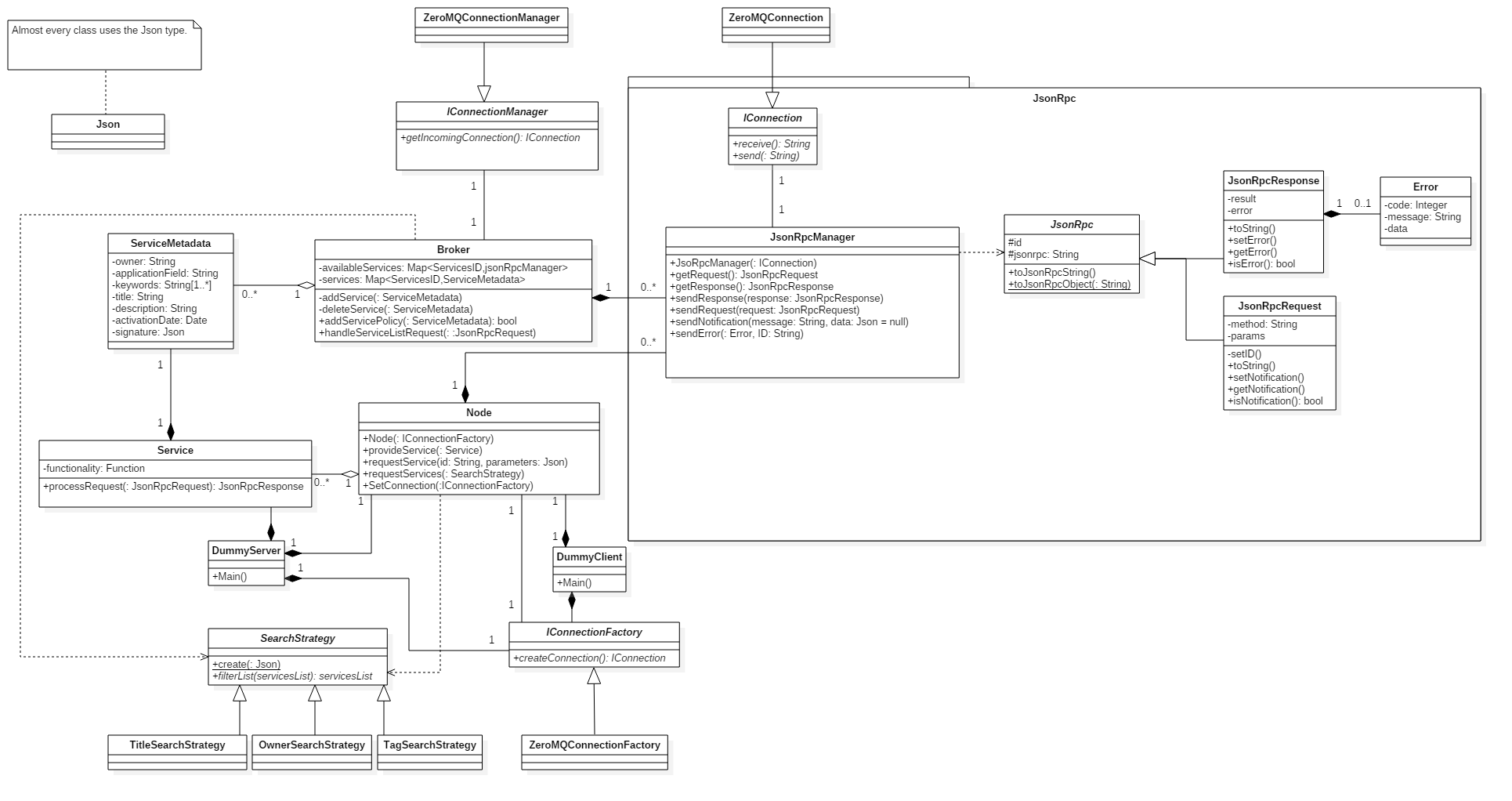
In questo diagramma vengono mostrati i moduli software in cui è diviso il sistema.

In particolare si può notare che il componente Node espone sia interfaccia dedicate alla funzione di Client e Server.

# Deployment Diagram



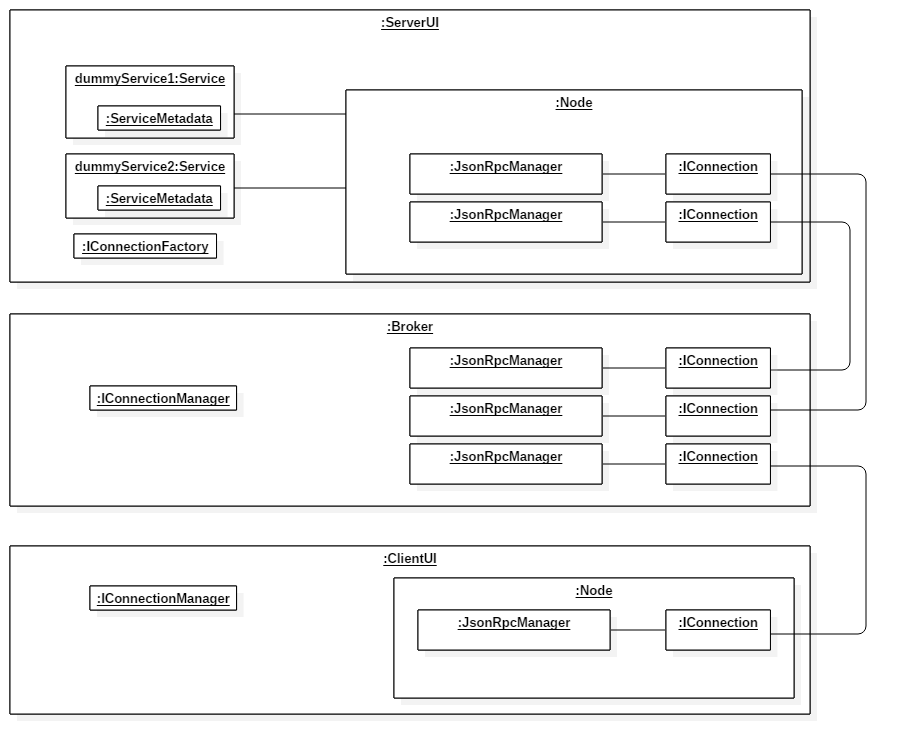
# Class Diagram



Nel class diagram sono rappresentate le classi del tema comune (rappresentata dal package JsonRpc) e del tema B. Di seguito, alcune precisazioni:

* Al fine di ottenere l’applicazione e la libreria JsonRpc agnostiche rispetto al canale trasmissivo sono state previste le seguenti interfacce:
  + IConnection: è un’interfaccia che si occupa di gestire una comunicazione già instaurata. La presenza di questa interfaccia è necessaria affinchè i metodi di ricezione e invio di risposte e richieste siano da implementare indipendentemente dalla libreria di comunicazione.
  + IConnectionManager: è l’interfaccia attraverso la quale il Broker ottiene per ogni connessione in arrivo una IConnection.
  + IConnctionFactory: è l’interfaccia attraverso la quale il Node può instaurare una connessione con il Broker.
* Per poter consentire la ricerca da parte del client, dei servizi disponibili nel sistema, è stata prevista nel modello la classe astratta *SearchStategy* che permette l’indicazione di criteri di ricerca*.*
* Sebbene nella specifica JSON-RPC gli oggetti JSON-RPC Notification e JSON-RPC Error siano trattati, non abbiamo ritenuto opportuno prevedere le classi JsonRpcNotification e JsonRpcError poiché sono dei casi particolari rispettivamente di JsonRpcRequest e JsonRpcResponse. Per rispettare comunque la specifica sono stati previsti dei metodi per la realizzazione dei casi particolari sopracitati.
* JsonRpcManager è una classe che si occupa dello scambio di messaggi, secondo lo standard JSON-RPC, per una determinata IConnection.
* addServicePolicy è un metodo astratto della classe Broker la quale concretizzazione consentirà, in funzione dei metadati passati come parametro, di stabilire se il servizio può essere aggiunto nel sistema.
* La cancellazione, la registrazione di un servizio e la richiesta della lista dei servizi disponibili nel sistema sono servizi predefiniti presenti all’interno del Broker.
* Le classi DummyClient e DummyServer sono programmi Client e Server d’esempio nel sistema.

# Object Diagram

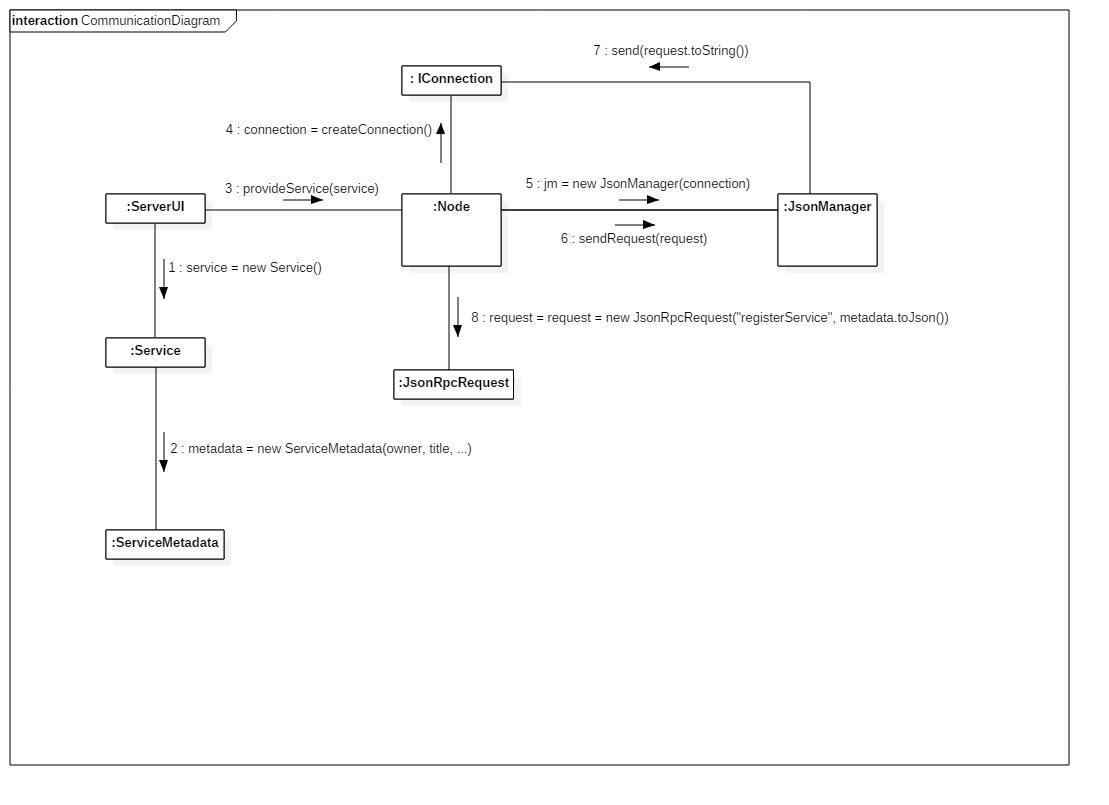


In questo diagramma è rappresentata una possibile configurazione degli oggetti.

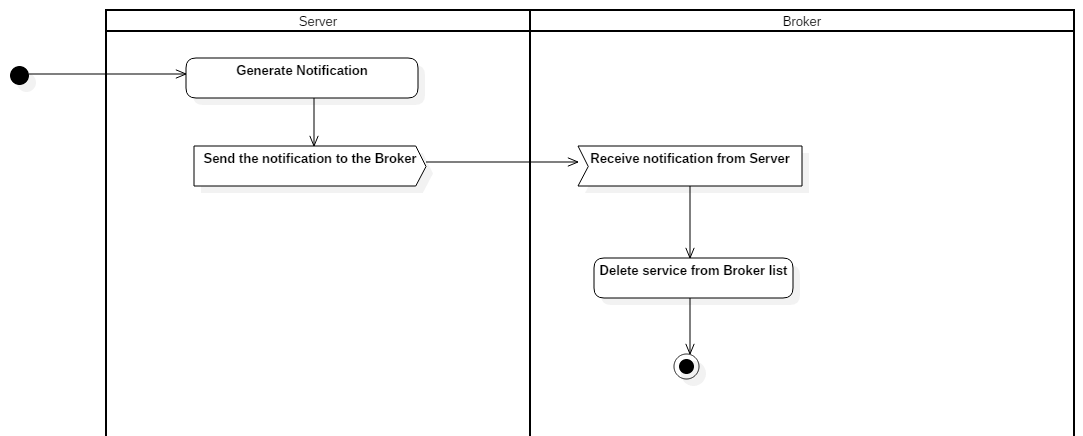
Nello specifico si possono notare tre entità principali:

* DummyServer è un fornitore di due servizi. La situazione attuale s
* DummyClient è un utilizzatore di servizi

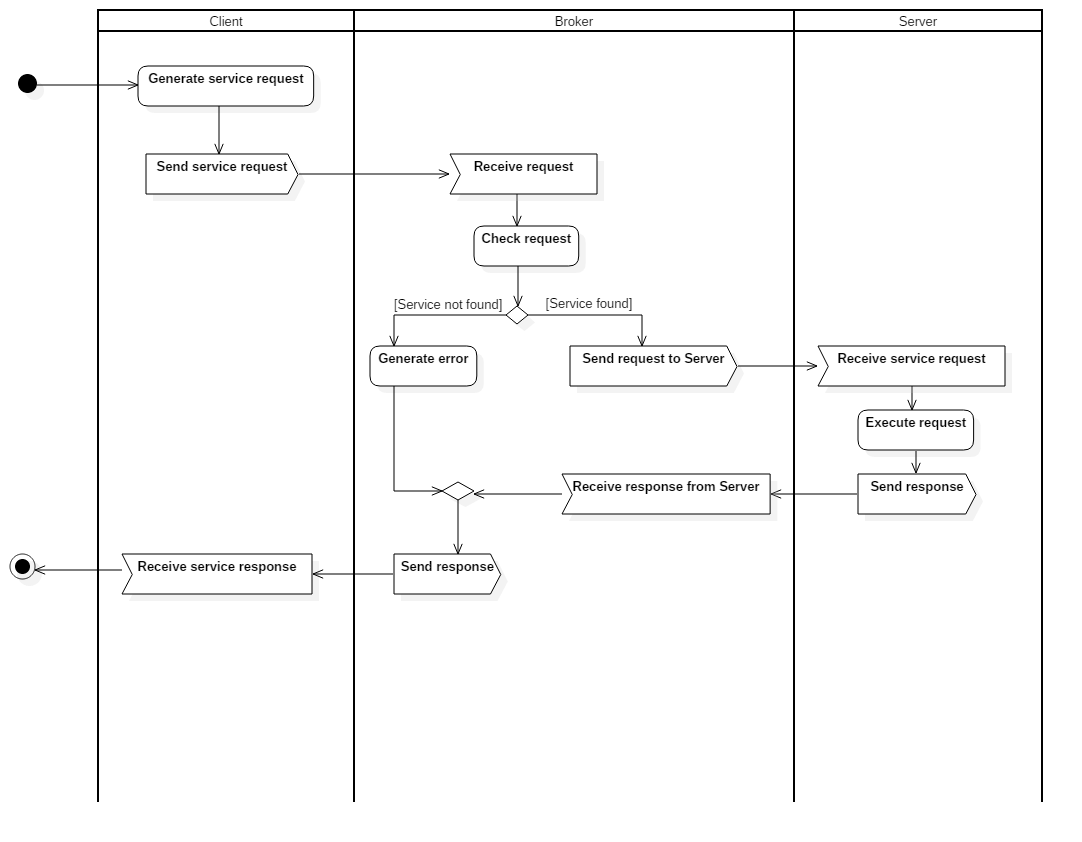
# Communication Diagram



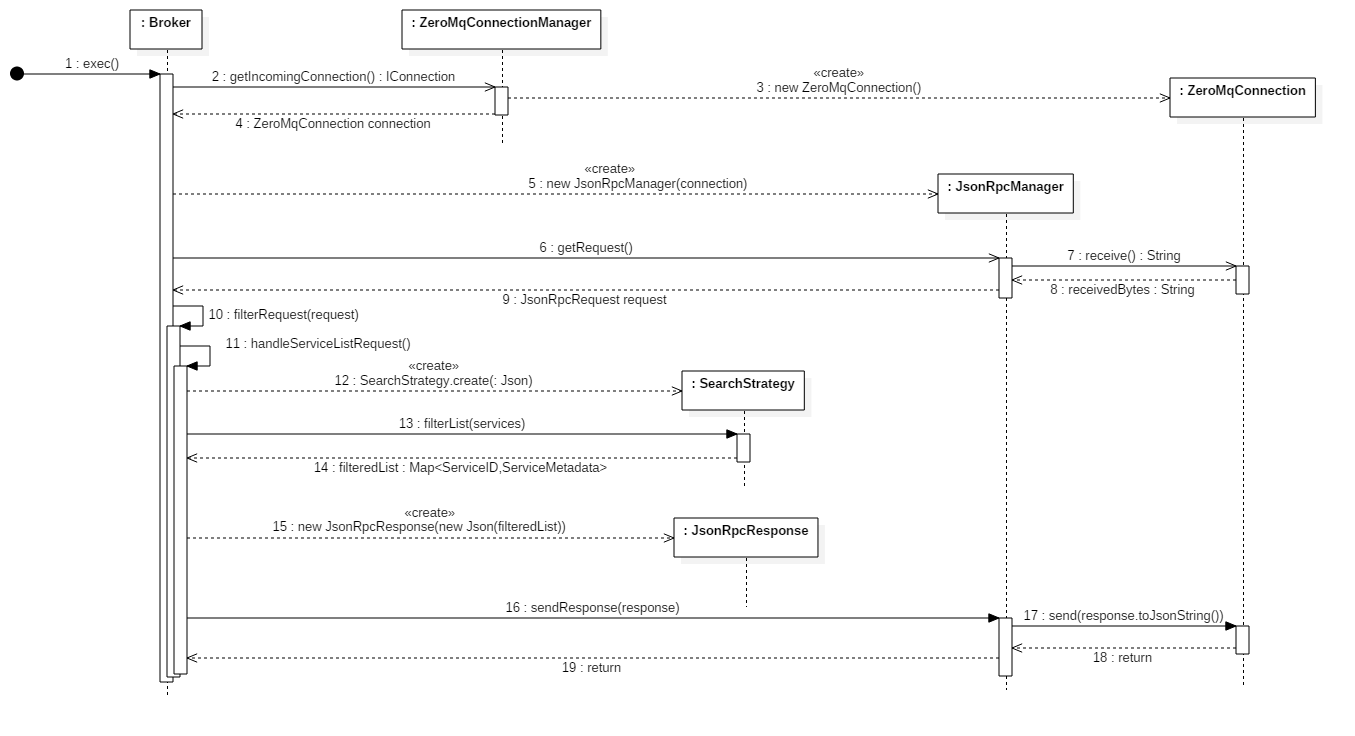
# Activity Diagram – Cancellazione di un servizio



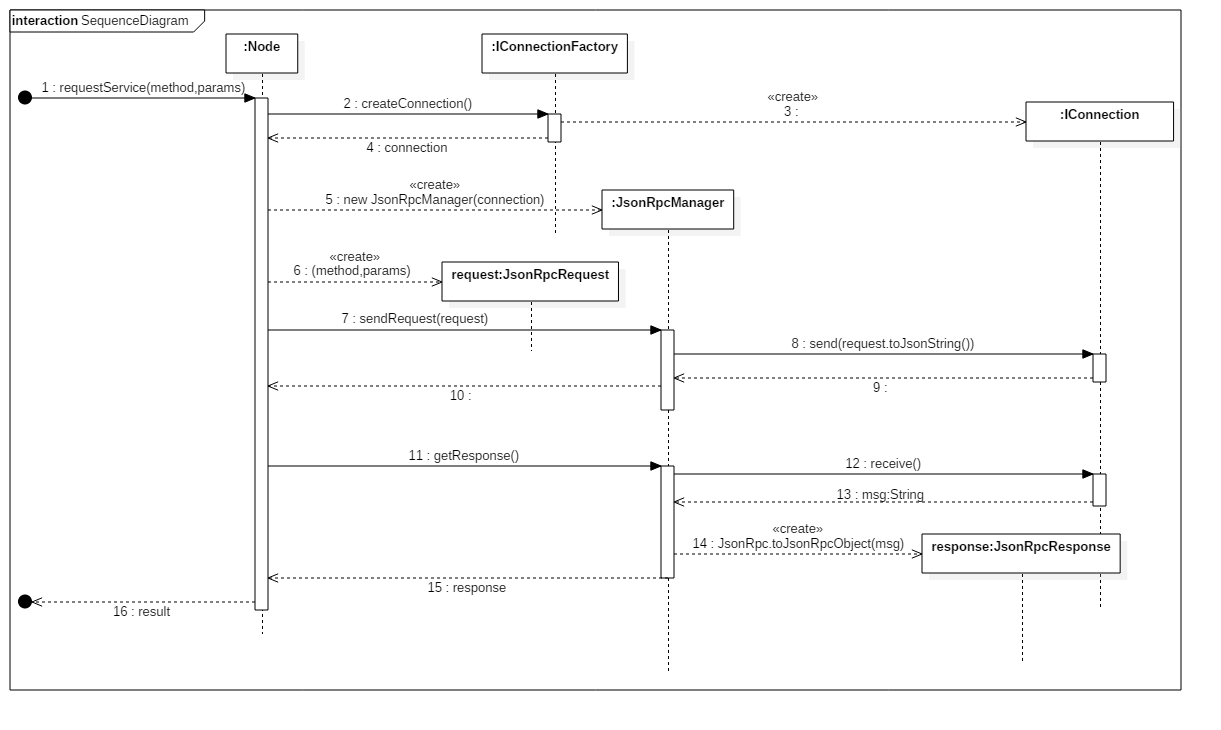
# Activity Diagram – Invocazione di un servizio



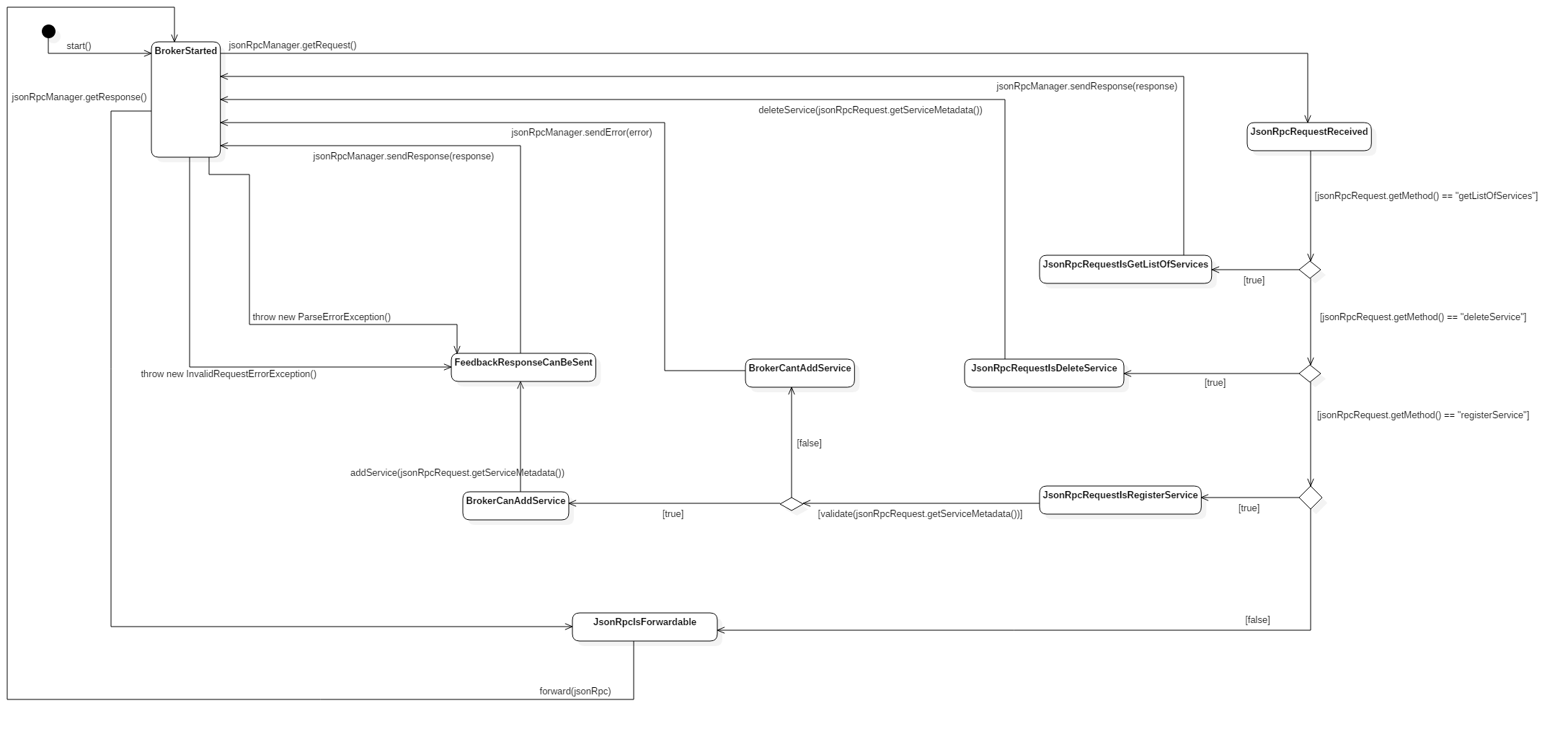
# Sequence Diagram – Richiesta della lista dei servizi



# Sequence Diagram – Invocazione di un servizio



# State Diagram – Broker

99

# State Diagram – Node

