

DESCRIZIONE ARCHITETTURALE E DESIGN DEL SOFTWARE CUPIDO

Lorenzo Belli, Marco Poletti, Federico Viscomi

1 Introduzione

Il presente documento descrive l'architettura software ed il design scelti per il software Cupido; esso è stato sviluppato, scelto ed approvato dal gruppo di lavoro firmatario del documento.

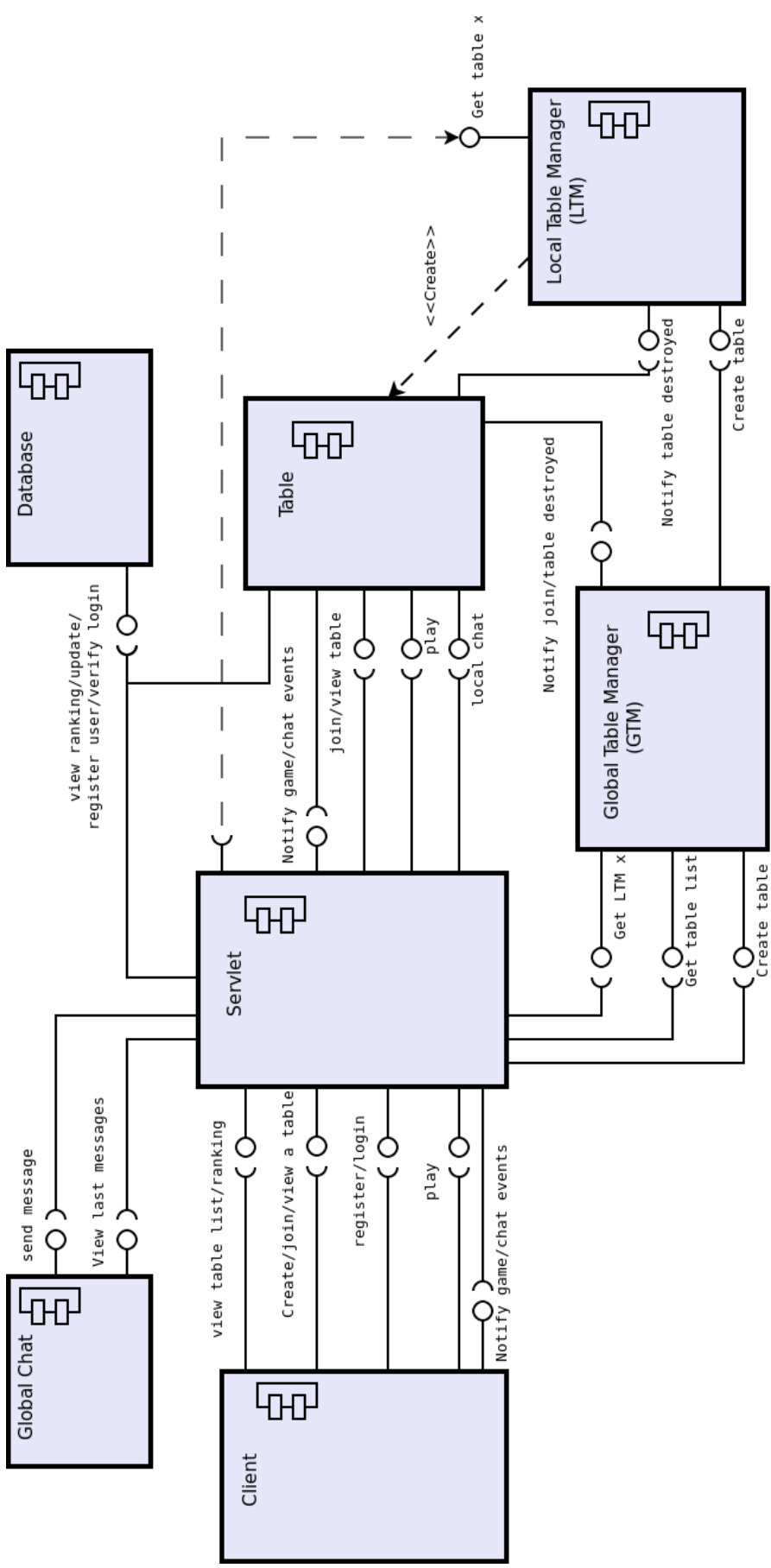
2 Referenze {TODO: Riferimenti?}

Per la stesura di questo documento si fa riferimento alla Specifica dei Requisiti Software disponibile in allegato nel file SRS.pdf. I requisiti che giustificano le scelte qui effettuate sono presenti nel documento SRS.pdf e saranno referenziati dalla sigla [RF xxxx]. Si assume che i lettori di questo documento conoscano già i Requisiti del Software qui descritto.

3 Abbreviazioni e sinonimi

- DB: database
- GTM: GlobalTableManager
- LTM: LocalTableManager
- Tavolo: sinonimo di partita {TODO: c'è confusione fra «tavolo» inteso come partita e «tavolo» come il componente SW che gestisce una partita.}

4 Architettura



L'utente si collega ad un web server tramite browser. Da qui il browser scarica ed esegue la componente client del software.

Client gestisce l'interfaccia grafica [RF 101-108] e gli input dell'utente, ed inoltra tutte le richieste alla servlet che risiede sul web-server a cui è collegato; Client inoltre resta in ascolto di notifiche relative ai cambiamenti di stato provenienti dalla servlet.

La servlet inoltra le richieste al tavolo che gestisce la partita. La servlet fornisce anche un'interfaccia di callback al tavolo, per poter essere notificata dei cambiamenti di stato che avvengono durante la partita. A sua volta la servlet inoltra queste notifiche al client.

LocalTableManager gestisce un insieme di tavoli, che implementano la logica di gioco. Ogni Table deve implementare la logica di gioco per i giocatori [RF 1001-1006][RF]{TODO}, la logica per gli spettatori [RF 1103-1104][RF 1201-1202] e l'utilizzo della Chat Locale [RF 903-905].

Per soddisfare i requisiti [RF 1601-1603] in ambito di scalabilità, è previsto che i LocalTableManager possano, al bisogno, essere replicati su più macchine fisiche in modo da distribuire il carico di lavoro. GlobalTableManager ha il compito di distribuire il carico di lavoro sui LocalTableManager secondo politiche interne.

Sempre per soddisfare [RF 1601-1603] prevediamo che anche i web-server possano essere replicati, e come meccanismo di load-balancing scegliamo di utilizzare un DNS Name Server{TODO: Togliere Name Server?} in grado di cambiare dinamicamente l'associazione dominio-IP (componente non mostrato nel diagramma).

Per creare un nuovo tavolo [RF 801] la servlet deve usare l'interfaccia 'Create Table' di GTM, che seleziona l'LTM più appropriato, e restituisce un oggetto remoto che si riferisce al tavolo creato.

GTM mantiene la lista di tutti i tavoli con partite in corso o non ancora iniziate; per ottenere tale lista [RF 703] la servlet deve richiederla al GTM attraverso l'interfaccia Get Table List. Per mantenere aggiornata la lista dei tavoli, GTM ottiene notifiche da Table attraverso l'interfaccia Notify Join/Table Destroyed.

La lista dei tavoli restituita da GMT contiene anche gli identificatori di LTM e TABLE, che sono inoltrati al client.

Quando un client vuole assistere o partecipare ad una partita [RF 802-803] fornisce alla servlet gli identificatori del LTM e del Table desiderato. La servlet ottiene un riferimento al LTM scelto tramite l'interfaccia 'Get LTM x' del GTM. In modo simile, ottiene un riferimento al tavolo remoto con l'interfaccia 'Get Table X' del LTM prescelto.

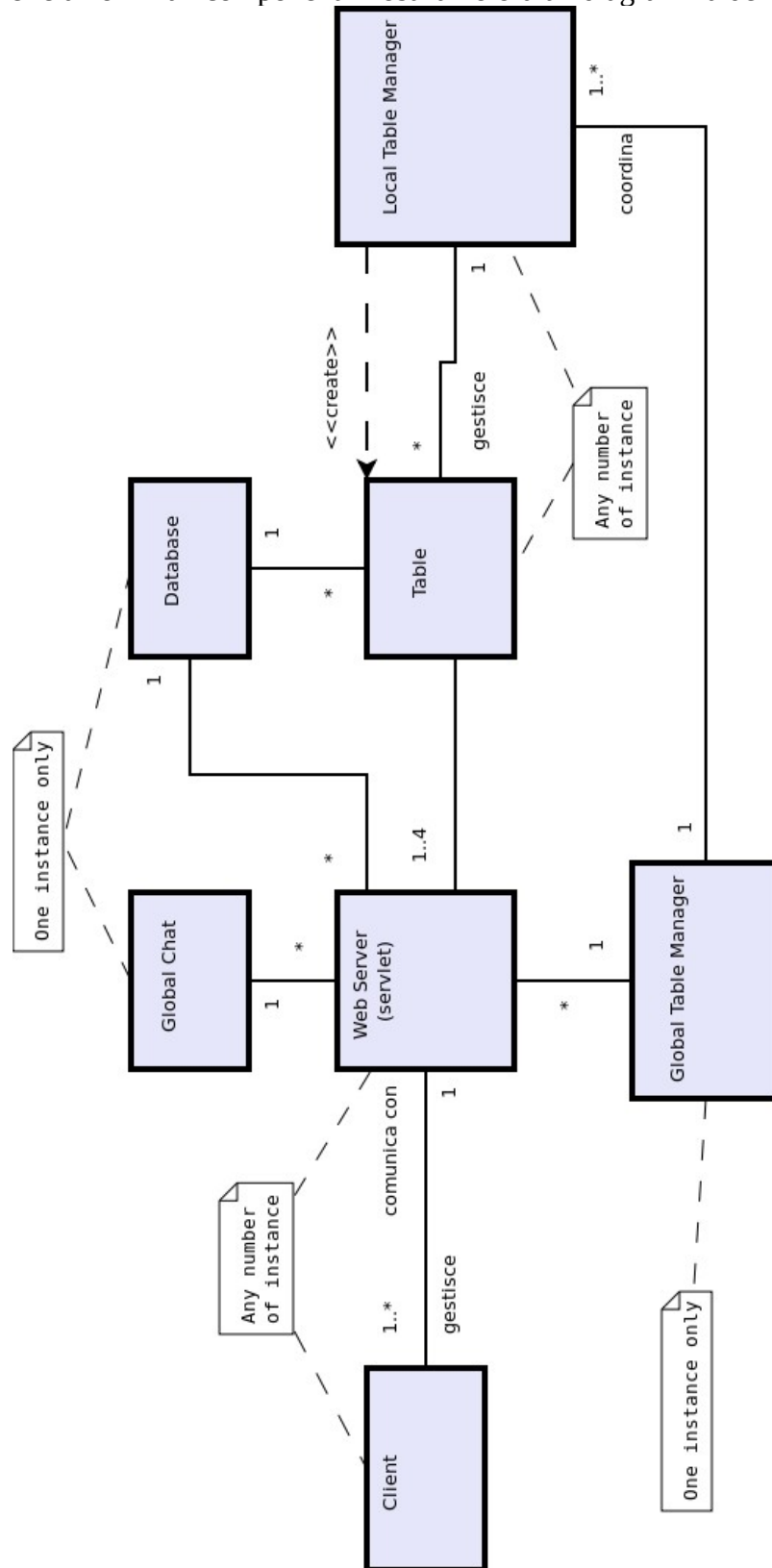
Database è un componente che racchiude la lista di tutti gli utenti registrati e fornisce un'interfaccia per l'accesso ai dati. Database deve anche mantenere la classifica degli utenti come descritto in [RF 709-712].

La servlet ed il tavolo accedono direttamente a Database manager per registrare un utente [RF 602-603], effettuare il login, visualizzare la classifica [RF 805] ed aggiornare la classifica [RF 709].

Global Chat è un componente indipendente che implementa la chat globale [RF 704-708]. Esso è condiviso fra tutte le servlet di tutti i web-server, e mantiene una lista degli ultimi messaggi inviati, per soddisfare [RF 705]. Client effettua un polling sulla servlet per ottenere gli ultimi messaggi inviati nella chat; a sua volta la servlet inoltra queste richieste al componente GlobalChat. Questo metodo di

interazione deve rispettare il vincolo temporale [RF 708].

Per chiarire meglio le relazioni fra i componenti mostriamo ora un diagramma della classi.

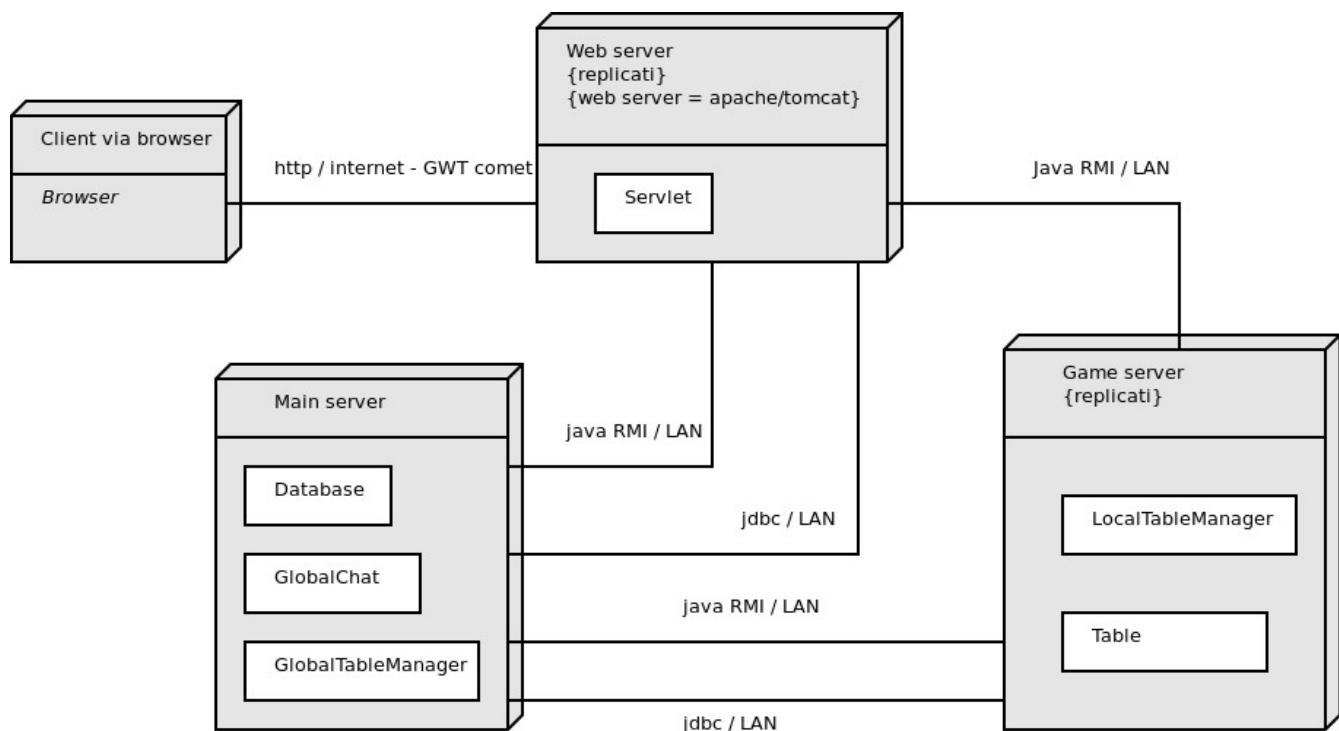


5 Design

Saranno qui esposte le scelte effettuate in ambito di design che serviranno a guidare il lavoro degli sviluppatori.

La comunicazione tramite client e servlet avverrà tramite protocollo HTTP, e per le notifiche dalla servlet al client useremo GWT-Comet [<http://code.google.com/p/gwt-comet>]. Il web server scelto è apache-tomcat 5.5 [<http://tomcat.apache.org/>]. La comunicazione tra Servlet, GTM, LTM e GlobalChat avverrà tramite java-rmi. Anche le notifiche che riceverà la servlet da Table saranno inviate in modo asincrono tramite java-rmi{TODO: ma non sono sincrone?}. Il database utilizzato sarà MySQL [<http://dev.mysql.com>].

Mostriamo un deployment diagram per chiarire come intendiamo distribuire i componenti sulle macchine fisiche.

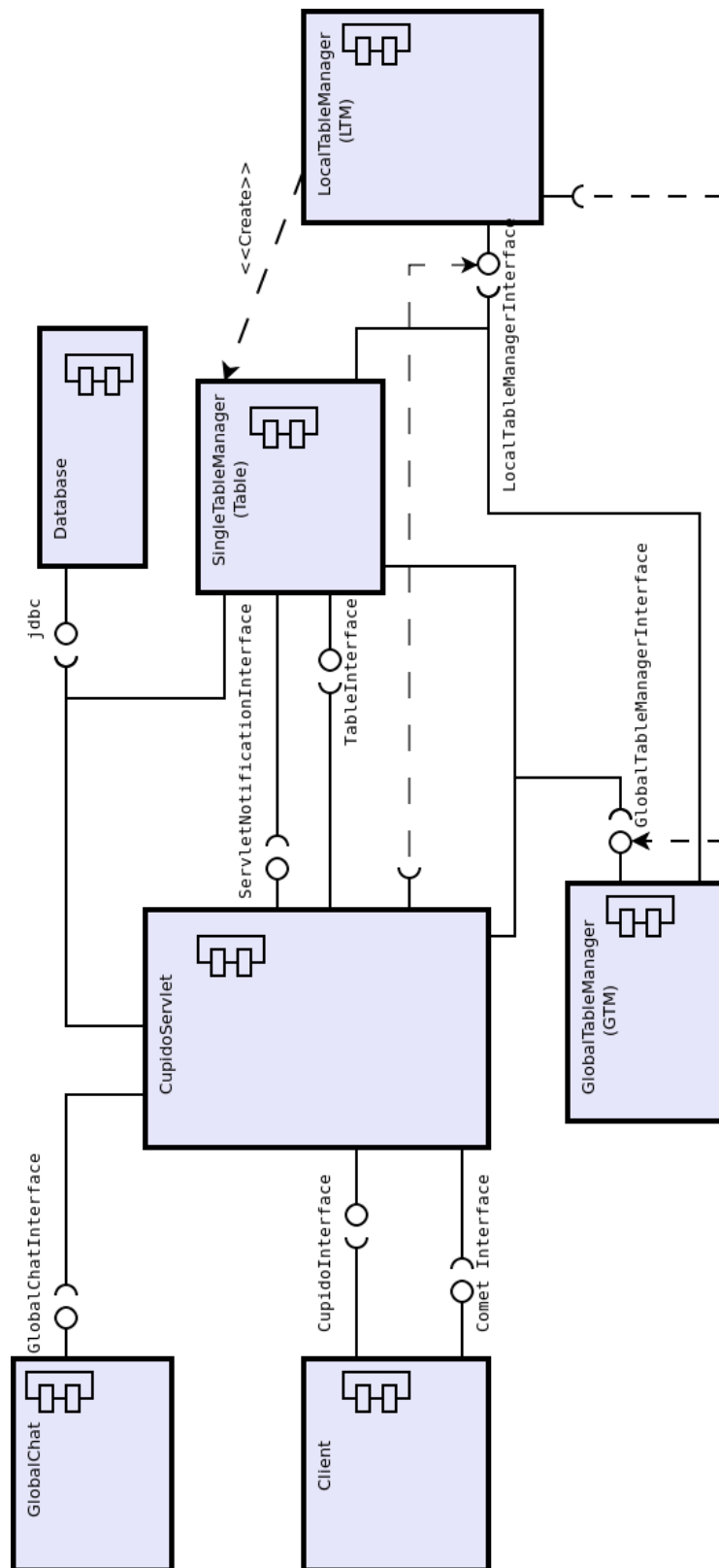


Ogni istanza di Web Server e Game Server sarà eseguita su un server fisico dedicato. I componenti Database, GlobalChat e GlobalTableManager potrebbero essere posizionati su nodi diversi della rete, ma noi li disporremo sulla medesima macchina.

Dopo aver deciso lo schema di deployment sono state definite tutte le interfacce dei componenti; le interfacce contengono i prototipi di tutti i metodi da implementare.

Mostriamo un Component diagram che specifica quali interfacce sono state create.

Per la specifica scelta delle interfacce si rimanda direttamente alla documentazione del codice sorgente.



Le interfacce CupidoInterface e CometInterface sono state pensate per minimizzare il numero di comunicazioni tra Client e CupidoServlet, e per minimizzare il calcolo da far svolgere a Client. Le interfacce di GTM ed LTM sono state unificate in due uniche interfacce, rispettivamente chiamate GlobalTableManagerInterface e LocalTableManagerInterface