**CAPITOLO 14: CONCLUSIONI E SVILUPPI FUTURI**

Lo scopo della tesi era quello di rintracciare una serie di strumenti al fine di creare una infrastruttura per permettere ad uno studente del corso di Architetture Software di testare quanto appreso a lezione. Diverse sono state le idee proposte come Vagrant, Spring e Maven. E’ stata ideata un’applicazione d’esempio, ispirandosi a Car2Go, per poi implementarla mediante tutte le tecnologie affrontate nel corso di Architetture Software.

Far sviluppare allo studente una applicazione permette al discente di acquisire maggiore consapevolezza di quanto appreso in classe. Lavorare in piccoli team eterogenei rappresenta un importante momento di confronto che favorisce l’apprendimento. Il poter concretizzare gli argomenti affrontati nel corso ha l’importante ripercussione di creare un maggior coinvolgimento degli studenti.

Un possibile sviluppo futuro per tale lavoro potrebbe essere l’utilizzo di Vert.x. Tale strumento è una piattaforma per costruire applicazioni scalabili e real-time sulla Java Virtual Machine. Attraverso un Event Bus distribuito che permette l’interazione asincrona di diversi componenti e un sistema modulare, è possibile realizzare applicazioni con diversi linguaggi come JavaScript, Java, Ruby, Groovy, Python e altri non supportati ufficialmente. Vert.x fornisce inoltre il supporto a tecnologie come WebSocket e SockJS per realizzare applicazioni real-time con notifiche push lato server. Ogni componente che viene rilasciato su Vert.x è un Verticle (vertice), ovvero una singola unità che permette di soddisfare una particolare richiesta e che può dialogare con altre componenti attraverso l’Event Bus.

Vert.x è una piattaforma davvero interessante, sia per il paradigma architetturale che per le feature esposte. Una delle sue maggiori potenzialità risiede nel fatto di poter realizzare componenti disaccoppiati, che comunicano attraverso una canale asincrono e soprattutto che possono essere implementati in differenti linguaggi. Se si ha la necessità di utilizzare Ruby per un component, magari solo perché solo in quel linguaggi si dispone di una particolare libreria, la restante parte dell’applicazione potrebbe essere tranquillamente realizzata in un altro linguaggio senza doversi preoccupare di interazioni specifiche. Anche la possibilità di realizzare applicazioni con WebSocket o SockJS permette di affrontare una serie di scenari che risultano molto più complicati con framework di vecchia generazione. Infine è doveroso specificare che Vert.x è altamente scalabile e gestisce al suo interno rinomate piattaforme come Hazelcast per la distribuzione su cluster e Netty per la gestione del I/O.