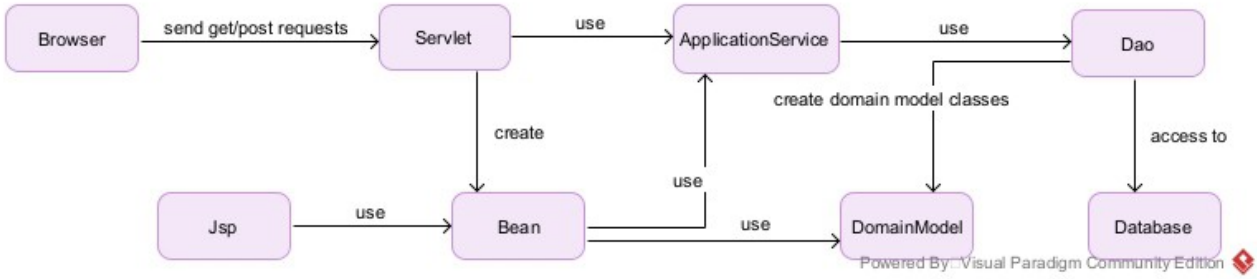
Breve relazione
per il corso di

Sistemi Distribuiti
parte relativa alle tecnologie web

Stefano Agarbati

Riproduzione molto parziale di un'applicazione web per la definizione di test usata per acquisire familiarità con le principali tecnologie web quali HTML, CSS e Javascript per la parte client e Jsp, Servlet per la parte server.

Lo schema utilizzato per implementare le diverse funzionalità prevede una parte che si occupa di gestire la presentazione delle informazioni e le interazioni con l'utente, una parte lato server che si occupa della logica di business. Le interazioni con l'utente scatenano delle richieste http di tipo get o post le quali verranno ricevute ed elaborate da opportune servlet. La servlet può recuperare alcuni parametri passati attraverso la richiesta oppure memorizzati all'interno della sessione. Quindi procede alla creazione di un pojo il quale verrà inserito all'interno della richiesta come attributo per poi ridirigere la richiesta stessa ad una jsp. La jsp genera l'interfaccia utente dinamicamente usando le informazioni fornite dal pojo il quale recupera i dati a partire da application services. Gli application services incapsulano la eventuale logica di business e accedono ad una base dati, sorgente delle informazioni necessarie alla realizzazione della logica aziendale.
In alcuni casi è la servlet stessa ad utilizzare application services per la logica aziendale.
Questo schema è tipico del modello 2 di java enterprise.

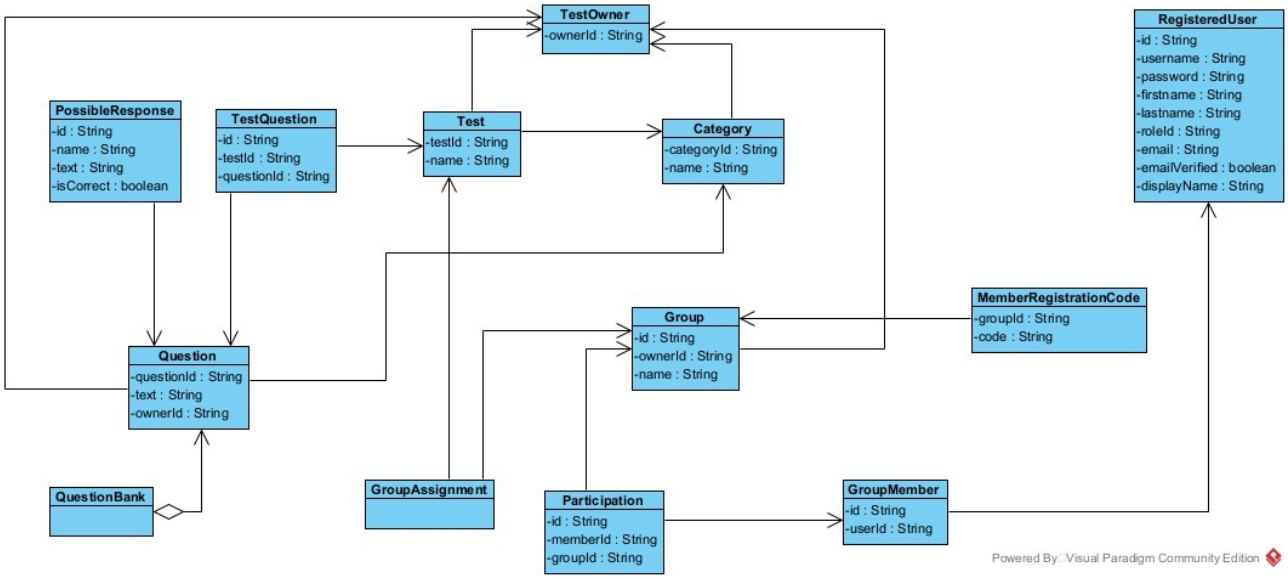


Per poter fare qualsiasi cosa, gli utenti devono prima registrarsi. Un utente può registrarsi come educator oppure come taker. L'educator è colui che formula i test mentre il taker è colui che sostiene un test definito da un educator. L'educator può registrarsi attraverso una username ed una password. Il taker può essere registrato direttamente da un educator oppure può registrarsi autonomamente attraverso un codice fornito da un educator. In ogni caso l'accesso successivo alla registrazione avverrà per mezzo di una username ed una password. Dopo l'accesso, un educator, può definire un test con le relative domande e risposte, può creare nuovi gruppi ed aggiungere a tali gruppi dei membri, assegnare un test ad un gruppo oppure ad un link a cui è possibile accedere per sostenere il test senza la necessità , per un taker, di dover appartenere ad un gruppo specifico. L'applicazione originale consente moltissime altre funzioni.

Ho implementato solo alcune funzionalità fra cui quelle di:

- registrazione
- accesso
- definizione di test
- definizione di domande
- assegnamento di un test ad un gruppo
- definizione di gruppi
- definizione di categorie
- e poche altre.....

Il domain model è completamente anemico ed è molto simile ad uno schema di una base di dati relazionale normalizzato (o quasi)



Questo esercizio, a mio parere, è stato di una certa utilità. Mi ha permesso di apprendere le principali tecnologie utilizzate nella pratica per la realizzazione di applicazioni web. Le tecnologie lato client quali html, css, javascript sono sostanzialmente onnipresenti in tutti i front-end sebbene la tendenza attuale è quella di farsi aiutare da alcuni noti framework come Jquery, Angular, React, VueJs …... Ho potuto osservare che la definizione della parte frontale di una applicazione web richiede moltissimo tempo. E' necessario descrivere ogni più piccolo dettaglio, dal layout degli elementi (ovvero la disposizione dei vari elementi all'interno di una pagina), ai colori, alle dimensioni (degli elementi, dei caratteri, dei bordi.....), alle transizioni ed animazioni, alla gestione degli eventi generati dai componenti a seguito dell'interazione con l'utente e così via. Anche la parte server possiede certe sue complessità. Necessità della modellazione di tutte quelle informazioni trattate dall'applicazione ovvero della definizione di un modello del dominio, della definizione dei dati che dovranno essere ospitati nella base dati ovvero della definizione dello schema per la base dati (ad es. tabelle nel caso relazionale), della definizione delle funzionalità di business implementate al di sopra del modello del dominio, la definizione delle pagine generate dinamicamente da restituire al client ovvero jsp e poi la definizione dei componenti di interfaccia quali servlet, servizi web ecc... attraverso i quali esporre e rendere accessibili le funzionalità core dell'applicazione. La definizione delle pagine jsp è piuttosto ostica e richiede molto tempo soprattutto perché eventuali modifiche non sono immediatamente visibili, ogni volta è necessario riavviare il server (processo avido di tempo e che richiede molta pazienza). La parte server potrebbe essere basata su tecnologie diverse. Oltre java ho notato che ne esistono altre. Correntemente sembra essere molto usato NodeJs (un ambiente che consente l'esecuzione di codice javascript e la realizzazione di server web e applicazioni web).
La tendenza attuale, probabilmente per motivi di scalabilità, è quella di costruire pagine web direttamente lato client manipolando il dom con javascript e usando le informazioni restituite dal server in formato generalmente json. Molte interazioni vengono gestite in maniera asincrona secondo il modello AJAX: il front-end invia al server una richiesta http post o get ed il server risponde inviando informazioni in formato json usate poi da uno script javascript per manipolare la pagina e visualizzare i dati ottenuti.

Concludo questa brevissima relazione riportando la positività dell'esperienza fatta in relazione alle tecnologie web ed una possible volontà ad approfondirne alcuni aspetti e tecnologie.