



**Partecipanti**:

|  |  |
| --- | --- |
| **Nome** | **Matricola** |
| Valenti Andrea | 0522500403 |
| Papaleo Giuseppe | 0522500465 |
| Tammaro Ruggero |  |

**Scelte effettuate e valutazione delle possibili alternative :**

**-Realiazzione DBMS**

Per la gestione del DB vi era una indecisione iniziale sull’ utilizzo di JDBC o Hibernate. JDBC è un’astrazione di basso livello che permette di fare tutto ciò che si fa con SQL. Il suo svantaggio è quello di dover realizzare tutto manualmente. Hibernate, invece, è un ORM(Object-Relation Mapping). Il suo scopo è quello di rendere più veloce e facile la creazione di modelli ricchi e interconnessi, esso è ottimo per la realizzazione di un’applicazione che non abbia un modello di dominio troppo complicato.

Da un’accurata analisi del problema si è giunti alla decisione di abbandonare l’ alternativa Hibernate poichè si è deciso di riutilizzare l’infrastruttura del sistema preesistente per cui l’utilizzo di un ORM avrebbe comportato una ridondanza di classi che erano gia esistenti e che Hibernate avrebbe creato durante la mappatura delle tabelle del DB.

Per tanto la scelta è ricaduta su JDBC e la mappatura delle tabelle in oggetti , così come l’accesso al DB tramite le opportune query, sono state effettuate manualmente.

Di fondamentale inmportanza è stata la scelta dell’ utilizzo del connection pooling e il Java Name Interface. Generalmente stabilire una connessione a un server database comporta passaggi che richiedono molto tempo. Durante l'esecuzione dell'applicazione, saranno aperte e chiuse più volte connessioni identiche. Per ridurre al minimo il costo dell'apertura delle connessioni è stata adottata una tecnica di ottimizzazione denominata pool di connessioni.

Il pool di connessioni riduce il numero di volte in cui è necessario aprire nuove connessioni

che gestisce mantenendo attivo un set di connessioni attive per ogni configurazione di connessione. Quando un utente chiama “Open” su una connessione, il pool verifica la presenza di una connessione disponibile. Se è disponibile una connessione, il pool la restituisce al chiamante invece di aprirne una nuova. Quando l'applicazione chiama “Close”, il pool restituisce la connessione al set di connessioni attive invece di chiuderla realmente. Una volta restituita al pool, la connessione può essere usata nuovamente nella successiva chiamata “Open”. Le connessioni vengono divise in pool in base alla stringa di connessione. Ed è qui che entra in gioco il java naming interface che permette ai client java di scoprire e ottenere dati e oggetti attraverso un nome(stringa di connessione).

**-Refactoring dell’ UI**

Per quanto riguarda l’ UI del software la decisione iniziale era stata quella di effettuare un reenginering dell’ UI tramite React.js, una libreria Javascript che permette la realizzazione di applicazioni web dinamiche a singola pagina (single-page web application) rendendo il loro sviluppo estremamente semplice e veloce. Inoltre tale libreria consente di creare sia applicazioni web da zero sia di essere integrata all'interno di un progetto già esistente.

Successivamente analizzando in modo accurato il sistema si è giunti alla conclusione di abbandonare l’idea precedente e di utilizzare JSP e Servlet per effettuare il re-enginering dell’ UI.

Introducendo JSP e Servlet è stato possibile riutilizzare il codice funzionale preesistente della vecchia applicazione. L’utilizzo di React.js com’ era stato inizialmente pensato doveva gestire oltre che l’ UI anche parte della logica. Ma come affermato in precedenza è stato deciso di salvare la logica preesistente del programma e pertanto tale scelta è stata accantonata ma non eliminata del tutto poichè React è stato comunque utilizzato per facilitare la costruzione dei componenti della View. Ciò ha reso possibile realizzare più velocemente una UI moderna e responsive senza perdere la logica del programma.

L’impiego delle servlet e le JSP ha permesso l’interfacciarsi con il backend dell’ applicazione e l’importare nell’ UI i valori restuiti dal DB.