Università degli Studi di Salerno

Corso di Gestione Progetti Software

A.A. 2012/2013

Alfonso Murolo

Giulio Franco

Linda di Geronimo

System Design Document

Versione 0.1

Storia delle revisioni

Versione	Data	Autori	Descrizione
0.1			

Riferimenti

Indice

1 1. Introduzione

L’obiettivo del documento SDD è quello di far fronte a tutti i problemi del sistema proposto fornendo un’architettura vantaggiosa, che possa rendere il sistema più efficiente possibile. A tale scopo si specificheranno quali caratteristiche e requisiti dovranno essere migliorati e definiti. Si forniranno così informazioni sulla configurazione hardware e software del sistema.

1.1 1.1 Scopo del sistema

Il sistema “At-silo” nasce con l'obiettivo di migliorare ed ottimizzare il servizio di asilo nido messo
a disposizione dall'università di Fisciano, inoltre una particolare caratteristica del sistema riguarda
la possibilità di interagire, in modo intuitivo, da parte del genitore sia per consultare informazioni personali riguardanti il proprio figlio, e sia per consultare i servizi messi a disposizione tramite l'accesso al sistema previa identificazione. A tale scopo il software richiede la presenza di requisiti minimi per prestazioni e portabilità. In particolare:

Adattabilità e Portabilità: avendo come requisito non funzionale l'Usabilità, le interfacce dovranno essere intuitive e semplici su ogni sistema software e hardware utilizzato, per questo motivo verranno utilizzate le tecnologie JDBC per quando riguarda la connettività ai server e la loro gestione tramite il package Servlet e il protocollo HTTP per lo scambio di informazioni via WEB tra client e server.

Tempi di Risposta e Prestazioni: avendo come requisito non funzionale le Performance, le interrogazioni al Database devono risultare rapide, inoltre il sistema dovrà favorire il requisito di Robustezza, ossia la resistenza all'immissione di input non validi. altri requisiti non funzionali da sviluppare in dettaglio saranno Affidabilità, Sicurezza e Modularità per gli utenti. Tenendo conto dei costi prefissati, il tutto si riassume in un unica tecnologia MySQL.

1.2 Obiettivi di design

Gli obiettivi di design rappresentano, in un prodotto software, le basi del successivo sviluppo del prodotto, perché, su di esse, si fondano le scelte prese durante la fase di implementazione.

Di seguito, sono presentati gli obiettivi del progetto @silo, ordinati secondo importanza decrescente, categorizzati in base agli utenti del sistema cui essi si applicano.

1.3 1.3 Trade Off

Interfaccia vs usabilità
L'interfaccia del prodotto At-silo è composta da oggetti molto comprensibili all'utente che vanno a chiarire immediatamente la propria funzione. L'interfaccia è composta da schede, pulsanti e varie etichette, associate all'oggetto, che fanno intendere la loro utilità.

Sicurezza vs Efficienza
La gestione della sicurezza viene affidata all'utilizzo del login iniziale in quanto va ad autenticare l'utente al quale sarà visualizzata solo la parte del software che gli appartiene, evitando così incongruenze di dati. Questa politica di permessi, permette di non appesantire eccessivamente il software ed è un buon compromesso tra sicurezza ed efficienza.

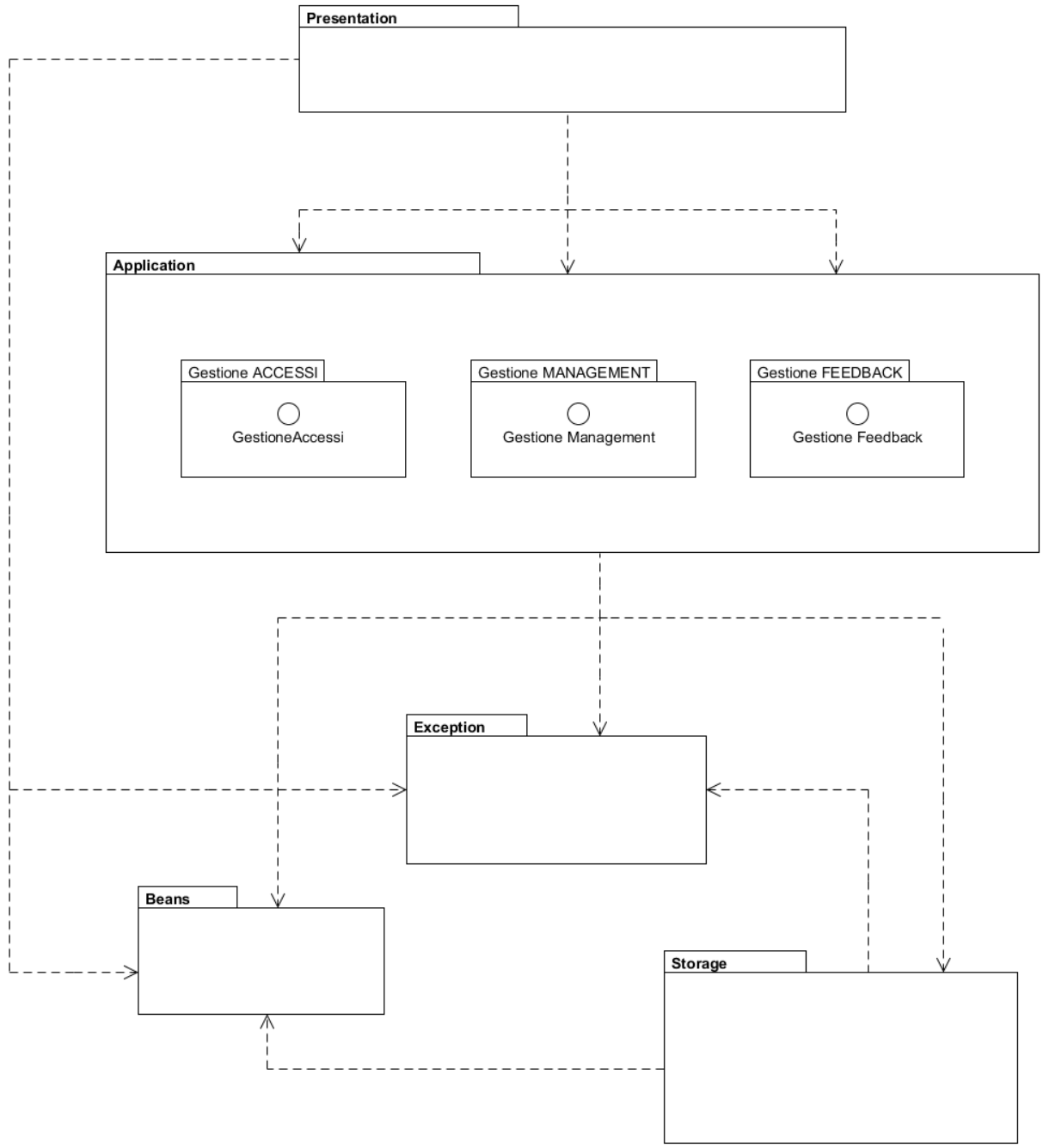
Comprensibilità vs Tempo
Il codice deve essere più comprensivo possibile in modo da poter essere interpretato da altri programmatori che non hanno partecipato al progetto. Una seconda motivazione è anche per non accrescere la difficoltà dello sviluppo nella fase di testing. Il codice sarà commentato in modo da migliorare la lettura delle righe di codice anche se l'aggiunta di commenti accrescerà il tempo necessario per completare l'implementazione.

Spazio di Memoria vs Velocità
Il prodotto dovrà memorizzare informazioni inerenti alle differenti entità riscontrate, essenzialmente il carico complessivo dei dati non influirà sulla velocità del sistema. Oltre modo le operazioni delle funzionalità implementate richiederanno un brevissimo tempo di risposta.
I costi per un disco fisso sono poco onerosi quindi si è scelto di dare più rilevanza alla velocità rispetto che alo spazio. La scelta di un DBMS rispecchia questa decisione in quanto I dati persistenti richiedono più spazio sul disco ma la velocità in lettura e in scrittura è molto alta.

Tempo di Rilascio vs Qualità
Le scadenze sono parte intrinseca del progetto, il nostro sistema garantirà oltre al rispetto delle date di consegna anche la qualità giusta delle funzionalità descritte e successivamente implementate.

2 Architettura proposta per il software

2.1 Decomposizione in sottosistemi

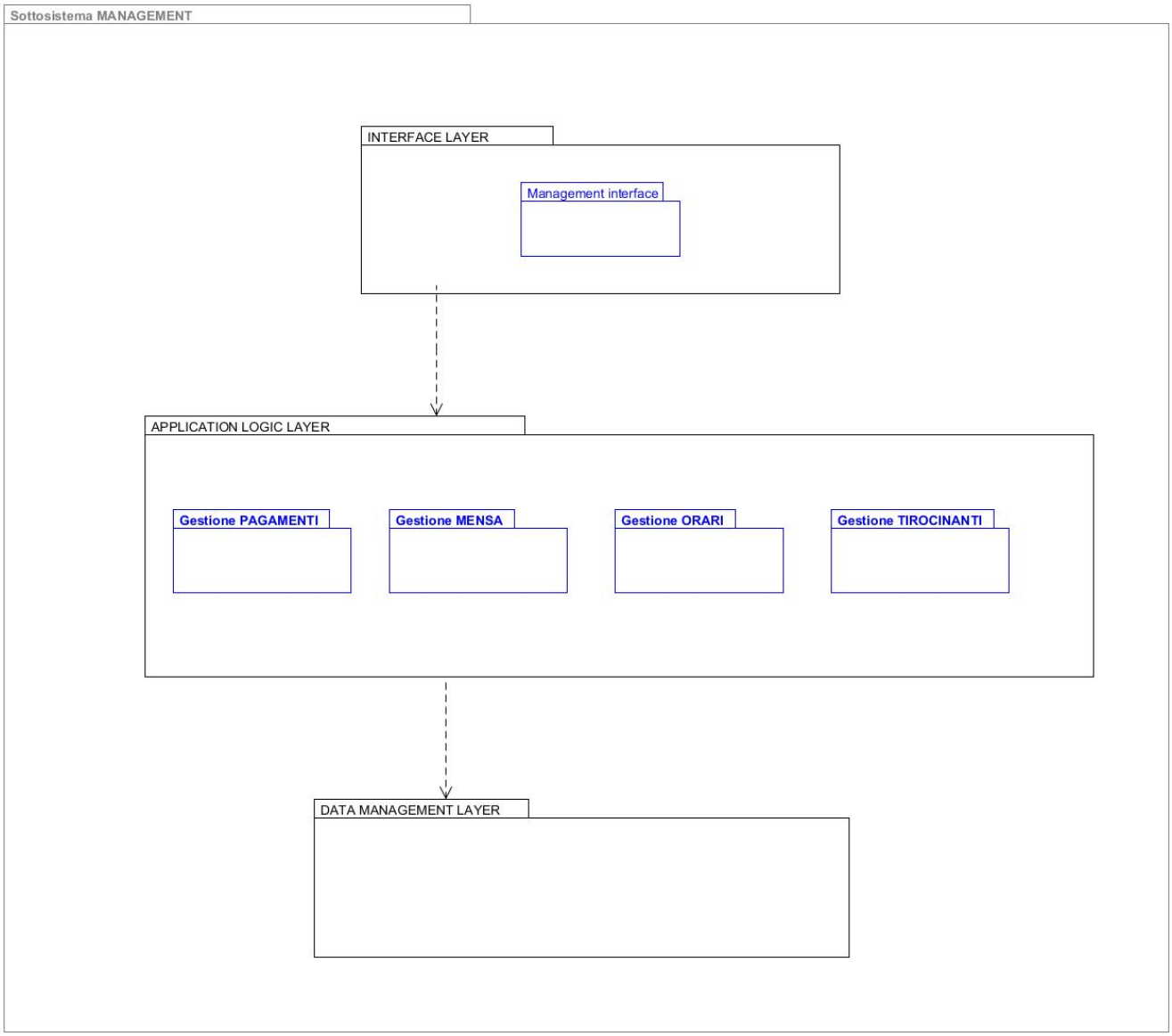


La decomposizione prevista per il sistema è composta da cinque sottosistemi che si occupano di gestirne aspetti e funzionalità differenti:

1. **Presentation**: raccoglie i sottosistemi adibiti alla gestione delle interfacce grafiche:
2. **Application**: si occupa della gestione della logica applicativa del sistema;
3. **Beans**: si occupa della gestione e dello scambio dei dati tra i sistemi;
4. **Storage**: sistema che gestisce ed immagazzina i dati persistenti:

5. **Exception**: gestione delle eccezioni del sistema.

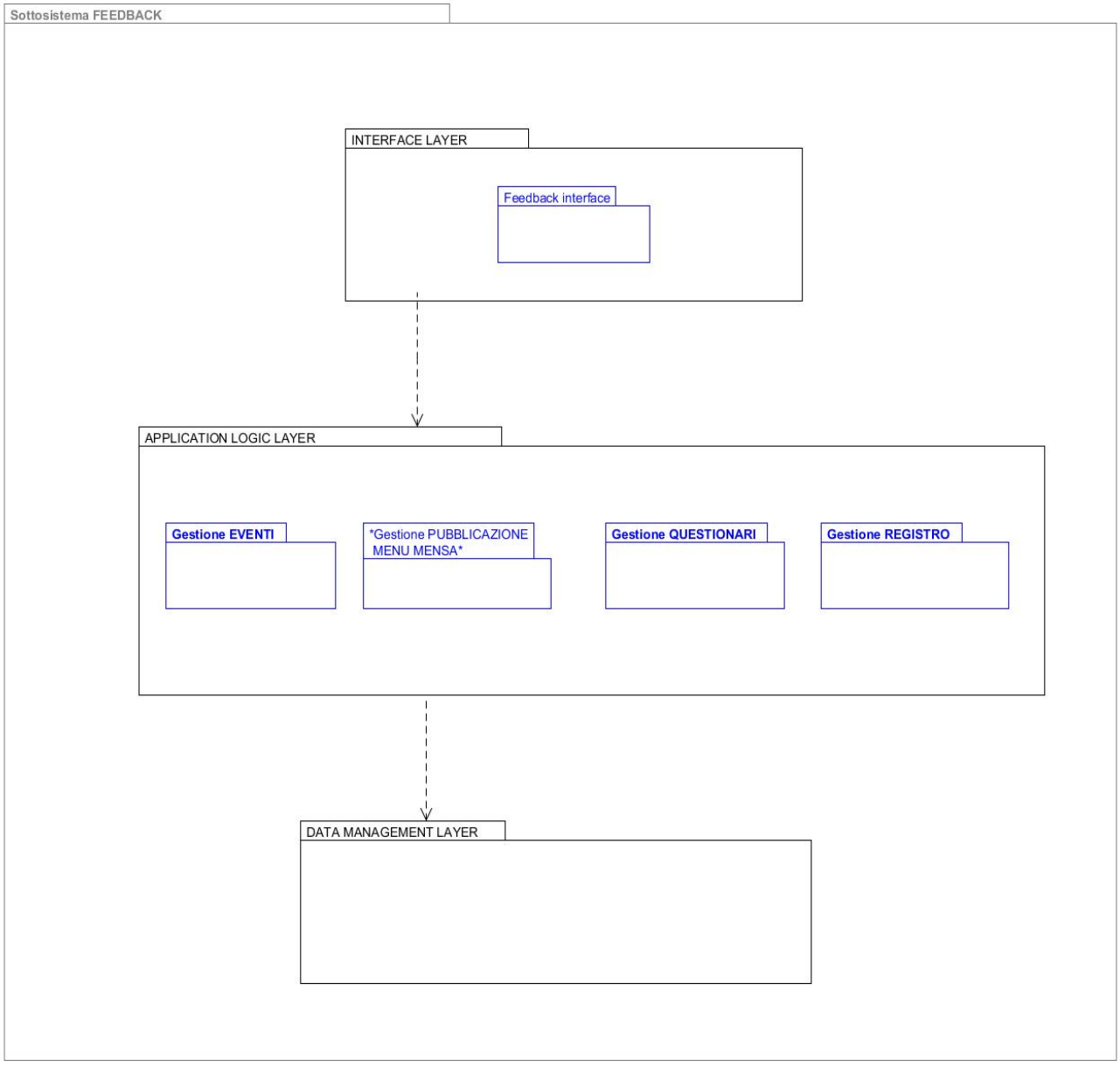
Sottosistema Management



- Interface layer
  - **Management Interface:** interfacce grafiche con cui i vari utenti gestori del servizio interagiscono con il sistema.
- Application Logic Layer
  - **Gestione Pagamenti:** modulo che si occupa di gestire il controllo e la visualizzazione dei pagamenti effettuati dagli utenti
  - **Gestione Mensa:** modulo che permette di richiedere ed effettuare modifiche al piano-pasto dei bambini.
  - **Gestione Orari:** modulo che permette di richiedere ed effettuare modifiche alla fascia oraria di utenza dei bambini.

- **Gestione Tirocinanti:** modulo che si occupa di gestire il periodo di tirocinio di studenti di Scienze della Formazione.

- Data Management layer
  Sottosistema Feedback

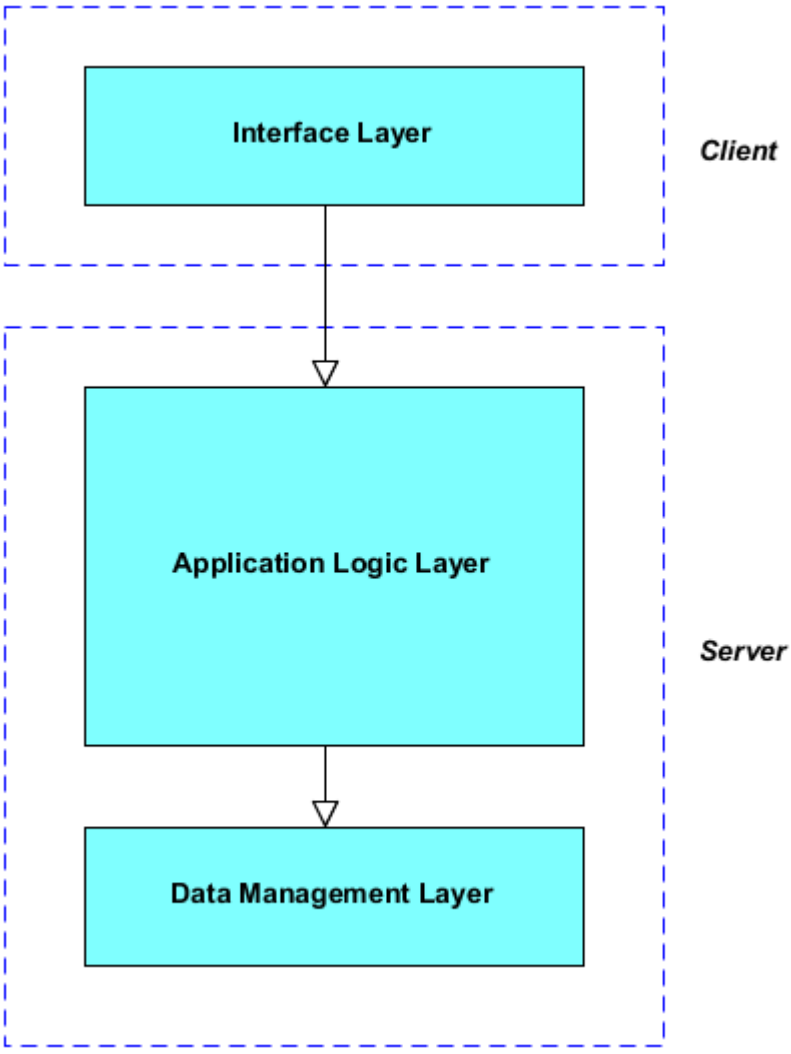


- Interface layer
  - **Feedback Interface:** interfacce grafiche con cui i vari utenti gestori del servizio interagiscono con il sistema.
- Application Logic Layer
  - **Gestione Eventi:** modulo che si occupa di gestire il controllo e la visualizzazione degli eventi creati dagli utenti

- **Gestione Pubblicazione Menu Mensa:** modulo che permette ad uno specifico utente di manutere il menu della mensa , inserendolo ,modificandolo e cancellandolo. Inoltre permette la visualizzazione del menu a tutti gli utenti.
- **Gestione Questionari:** modulo che permette di gestire i questionari, inserendoli, modificandoli per un particolare utente, compilarli e visualizzare le statistiche per altri utenti.
- **Gestione Registro:** modulo che si occupa di gestire in maniera computerizzata un registro standard cartaceo. Permette anche all'utente psico-pedagogo di visionare le attività svolte dall'educatore.

- Data Management layer

2.2 Layer e partizioni



L'organizzazione dei sottosistemi segue la logica **three-tier**. Si possono dunque individuare i seguenti layers:

- **Interface Layer**: lato client, è il layer caratterizzato dalle interfacce grafiche che permettono all'utente di interagire col server;
- **Application Logic Layer:** lato server, è il layer che contiene e gestisce le operazioni necessarie a compiere i servizi che l'utente richiede;

- **Data Management Layer**: lato server, è il layer che ospita e gestisce le entità persistenti (database e relativo DBMS).

La scelta è ricaduta sull'architettura three-tier allo scopo di gestire con facilità ed indipendentemente i sistemi di elaborazione dati e quelli relativi all'interfaccia grafica (una modifica al livello presentation non presenterà complicazioni in altri sistemi).

2.3 Topologia del sistema

<Introduzione alla topologia>

2.4 Implementazione del controllo software

2.4.1 Flusso di controllo esterno

Nel sistema @silo il flusso di controllo software sarà distribuito su più Client e più Server.

La logica applicativa sarà implementata sul Server e ai Client verranno mostrati i risultati avuti dall'esecuzione di alcune richieste ricevute dagli utenti.

Per quanto riguarda lo storage dei dati persistenti lo scambio di informazioni e di messaggi tra server e database management system (nel nostro caso mySql) verrà realizzato tramite l'implementazione di un dispatcher JDBC. Ciascuna funzionalità disponibile nel sistema @silo potrà richiamare il dispatcher JDBC per soddisfare le proprie richieste di memorizzazione ,ricerca e modifica dei dati.

2.4.2 Flusso di controllo interno

Il sistema @silo verrà utilizzato da una molteplicità di utenti, quindi esso dovrà garantire efficienza e coerenza per ogni utente di qualsiasi tipologia che si connetta al sistema.

Sui Client sarà eseguita un'applicazione che utilizza un flusso di controllo di tipo Event-driven, in cui il ciclo principale attende un evento esterno; Una volta che l'evento ha luogo, il programma invia all'handler le caratteristiche dell'evento ed esso viene spedito all'oggetto appropriato. Questo tipo di flusso di controllo ci permetterà di sfruttare una migliore efficienza delle interfacce grafiche, le quali risulteranno molto più flessibili ed usabili per gli utenti.

2.4.3 Controllo della concorrenza

2.4.4

Nel sistema @silo si possono avere diversi accessi agli oggetti in base all'utente che sta utilizzando il sistema.

Concorrenza: non sarà possibile effettuare operazioni su un oggetto mentre si stanno manipolando (salvando o modificando) istanze dello stesso tipo. Sarà invece possibile effettuare operazioni su oggetti di altro tipo. Gli utenti potranno concorrere accedendo allo stesso oggetto solo in lettura e non in scrittura in quanto questo potrebbe portare ad un'inconsistenza dei dati. E' necessario che l'accesso in scrittura sia fatto, dai diversi utenti, in sequenza e non in contemporanea, rispettando un certo ordine negli accessi. La gestione di questa concorrenza è gestita dal DBMS, ma anche dalla sincronizzazione degli eventi offerta dal linguaggio di programmazione scelto. Inoltre, ogni utente potrà svolgere una sola operazione per volta. Verrà implementato un controllo per ogni funzionalità del sistema, non sarà ammesso l'utilizzo concorrente di più funzioni all'interno dello stesso Client.

2.5 Mapping tra hardware e software

l sistema software @silo è distribuito su due nodi, Client e Server.
La configurazione HW necessaria per i Client è la presenza di una connessione ad internet nel caso di WebClient o di una connessione LAN nel caso di client Stand Alone, poiché il sistema @silo necessita di collegamenti remoti al server in cui è contenuto il sottosistema dello storage e quello delle application.

Inoltre i nodi Client necessitano di un sistema operativo e di una macchina virtuale Java (o JVM) per i clienti StandAlone oppure di un browser web per i WEBClient.

Le configurazioni SW dei Client sono dunque le seguenti:

- la macchina usata dai Dipendenti dell'asilo, su cui gira la versione 6.0 di Java Virtual Machine;
- la macchina usata dai Clienti su cui girerà un browser compatibile con HTML 5 (ad esempio Internet Explorer, Safari, Chrome, Opera e Firefox).

I nodi cliente permettono l'accesso al sistema tramite collegamenti remoti per lo scambio dei dati con in nodo server. I collegamenti sono gestiti da componenti off-the-shelf: HTTP è stato utilizzato per collegare il nodo “web” client al nostro server.

La configurazione HW minima del server comprende una connessione ad internet ed una alla rete locale insieme ad un hardware capace di immagazzinare grandi quantità di dati consistenti (HardDisk). Il nodo server, necessita inoltre di un software in grado di gestire le connessioni con più client e di una tecnologia che gestisca i dati persistenti. Per far ciò abbiamo deciso di utilizzare tecnologie già esistenti, quali Apache, che si occupa del lavoro del webServer e Mysql, un DMBS che gestisce i nostri dati persistenti.

Le configurazioni SW del Server sono:

- la macchina su cui risiederà il DMBS (MySql) con la relativa Base Dati del sistema.
- la macchina su cui risiederà il server Web (Apache) .

Le macchine potranno anche risiedere in diverse parti del server o su due server separati.

Ogni sottosistema risiede nei seguenti modi:

- **Presentation:** Risiede sul Client del Dipendente e del Cliente.
- **Application:** Risiede sul Server Web.
- **Beans:** Risiede sul Server con la base dati
- **Storage:** Risiede sul Server con la base dati
- **Exception:** Risiede sul Server con la base dati.

2.6 Gestione dei dati persistenti
2.6.1 Schema concettuale
2.6.2 Specifica delle entità
2.6.3 Progetto delle transazioni
2.7 Sicurezza e controllo degli accessi
2.7.1 Sicurezza
2.8 Performance del sistema
2.9 Condizioni di Boundary
2.9.1 Inizializzazione
2.9.2 Terminazione
2.9.3 Fallimento

3 Servizi dei sottosistemi

4 Glossario