Introduzione

1.1 Object Design Trade-Off

Abbiamo già discusso nelle precedenti fasi di analisi di quali saranno i criteri ed i compromessi su cui si baserà lo sviluppo del nostro sistema. Durante questa fase di Object Design, ci sono altri punti da analizzare, dunque altri compromessi da valutare, espressi di seguito:

Memoria / Estensibilità

Il sistema dovrà garantire, a discapito della memoria utilizzata, un'elevata estensibilità, così da poter essere flessibile a modifiche e variazioni, senza compromettere l'esperienza d'uso da parte dell'utente.

Affidabilità / Tempo di risposta

Come già sottolineato nei requisiti non funzionali (RADv3.2, punto 3.2.2) e nei Design Trade Off (SDD, punto 1.2.1), il sistema metterà al primo posto la correttezza delle informazioni piuttosto che il tempo di risposta, garantendo l'affidabilità dello stesso in ogni fase d'utilizzo.

Manutenzione / Performance

Il sistema verrà sviluppato prediligendo la portabilità e la manutenibilità del software, cosi da facilitare eventuali lavori di aggiornamento o estensione del sistema, a discapito però di una parte delle performance.

Costi / Robustezza

Non è garantita la massima robustezza del sistema per ridurre le ore uomo-lavoro e quindi non incorrere in un superamento del tetto massimo del budget. Quindi il sistema si avvarrà di un livello di sicurezza adatto a soddisfare le criticità basilari.

Trade-off	
Estensibilità	Memoria
Affidabilità	Tempo di risposta
Manutenzione	Performance
Costi	Robustezza

1.2 Componenti Off-the-Shelf

Per l'implementazione del nostro sistema, verranno utilizzate diverse componenti Off-the-Shelf.

Front End

Come già definito nei Requisiti non Funzionali (RAD, punto 3.2.2), gestiremo lo stile del sistema utilizzando HTML e CSS insieme con Bootstrap, un framework open-source contenente modelli di progettazione basati su HTML e CSS, i quali influiscono sulla tipografia e sulle varie componenti dell'interfaccia, estendendo inoltre le funzionalità di JavaScript.
Per il lato funzionale usufruiremo del framework di JavaScript JQuery, il quale permette di semplificare l'animazione del sistema e le chiamate AJAX, oltre che a poter interagire direttamente sul DOM del documento HTML.

Back End

Per la gestione del lato BackEnd, verranno utilizzati Java Enterprise come Web Container, ed Apache Tomcat come WebServer. Inoltre la funzionalità per l'invio delle e-mail verrà implementata tramite l'utilizzo della libreria JavaMail.

1.3 Design pattern

Per velocizzare lo sviluppo del sistema senza dover riscrivere del codice già esistente ogni volta, il team si avvarrà do alcuni design patterns che aiutano a risolvere dei problemi comuni riscontrati anche nel nostro progetto.

- **Singleton Pattern**
  - Il singleton è un design pattern di tipo creazionale, ed ha la funzione di garantire all'interno di un ambiente software un'unica istanza di una determinata classe, che nel caso del progetto EaseLease è la classe che si occuperà della connessione al DB.
- **Proxy Pattern**
  - Il proxy è un design pattern di tipo strutturale che fornisce un'interfaccia per oggetti che richiedono risorse e tempo per essere creati, e nel nostro caso verrà utilizzato per inviare delle e-mail agli utenti della piattaforma tramite la libreria "JavaMail".
- **DAO Pattern**
  - Il DAO è un design pattern di tipo architetturale per il basso livello di accesso ai dati e quindi per la gestione della persistenza. Nel nostro progetto saranno quindi delle classi con i relativi metodi che andranno a rappresentare le entità individuate nel DB, con lo scopo quindi di separare la logica di business e l'accesso diretto ai dati persistenti.

1.4 Linee Guida per la documentazione delle interfacce

Qui di seguito verranno stabilite le linee guida da seguire da parte degli sviluppatori del sistema al fine di essere consistenti per l'intero progetto e quindi facilitare la comprensione di ogni funzionalità del sistema all'intero team.

1.4.1 Classi e Interfacce Java

Lo standard che andremo ad utilizzare nella definizione di classi o interfacce Java è il code style definito da Google. Quindi ogni relativa documentazione dovrà seguire le relative linee guida.

1.4.2 Java Servlet Pages (JSP)

Per quanto riguarda le convenzioni adottate per la stesura delle Servlet e le relative JSP è stato adottato lo standard Oracle.

1.4.3 File JavaScript

Gli script javascript e la relativa documentazione seguiranno le linee guide definite da Google nel proprio CodeStyle.

1.4.4 Pagine HTML/CSS

Le pagine HTML devono essere conformi allo standard HTML5 e CSS3. Inoltre, il codice deve essere conforme alle linee guida descritte da Google.

1.4.6 Script SQL

Gli script SQL e il relativo Database seguirà le linee guida descritte nel libro "SQL Programming Style" di Joe Celko.

1.5 Definizioni, Acronimi, abbreviazioni

DOM = Document Object Model, rappresenta la strutturazione dei documenti come modello orientato ad oggetti
SDD = System Design Document
RAD = Requirement Analysis Document
CSS = Cascading Style Sheet
HTML = HyperText Markup Language
AJAX = Asynchronous JavaScript And XML
SQL = Structured Query Language
DAO = Data Access Object

1.6 Riferimenti

Google Coding Style: Java, Javascript, HTML/CSS
Oracle Coding Style: Servlet/JSP,
Simon Holywell: SQL