

PharmaÉlite



Corso di Ingegneria del Software

Odd - Object Design Document

Versione 2.7

**Partecipanti:**

|  |  |
| --- | --- |
| Nome | Matricola |
| Bozzoli Luigi | 0512105477 |
| Martucci Antonio | 0512105612 |
| Squitieri Lucio | 0512105180 |

**Indice**

**1.Introduzione**

1.1 Object design trade-offs

1.2 Componenti off-the-shelf

1.3 Documentazione dell’interfacce

1.3.1 Classi e interfacce java

1.3.2 Pagine lato server(JSP)

1.3.3 Pagine Html

1.3.4 Script Javascript

1.3.5 Fogli di stile CSS

1.3.6 Database SQL

1.4 Design pattern

1.4.1 MVC

1.4.2 Singleton

1.5 Definizioni, acronimi e abbreviazioni

**2. Packages**

2.1 Interface

2.2 View

2.3 Model

2.4 Controller

**3. Class interfaces**

**4. Class diagram**

**5. Glossario**

Object Design Document

1 Object design trade-offs

Durante la fase di Object Design sono sorti diversi compromessi di progettazione. Di seguito sono riportati i trade-off:

-Leggibilità vs tempo: L’obiettivo sarà scrivere codice che rispetti lo standard proposto da google per la programmazione con il linguaggio java, con l’aaggiunta di commenti per eventuali. Questo favorirà la leggibilità ed agevolare il processo di mantenimento e modifica del codice per gli sviluppatori che non avranno lavorato al progetto fin dall’inizio. Tuttavia, questo vantaggio aumenterà il tempo necessario per lo sviluppo e la realizzazione dell’intero sistema.

* **Affidabilità vs Tempo di risposta**: Il sistema sarà implementato in modo tale da preferire l’affidabilità al tempo di risposta in quanto si garantirà un controllo accurato dei dati in input per minimizzare errori.
* **Usabilità vs Funzionalità:** Il sistema sarà facilmente usufruibile dall'utente a discapito delle funzionalità offerte fa quest' ultimo.

1.2 Componenti off-the-shelf

Per l'implementazione del sistema utilizzeremo componenti self ossia componenti software già disponibili e utilizzati per facilitare la creazione del software. Il framework che utilizzeremo per la realizzazione delle interfacce grafiche delle pagine web è Bootstrap.

Per permettere all'interfaccia di rispondere alle azioni dell'utente e quindi velocizzare il sistema aggiungendo logica applicativa lato client si utilizzeranno JQuery, Javascript e Ajax.

Le pagine web visualizzate dall'utente si baseranno sui mock-UPS realizzati e consultabili nel punto (Aggiungere punto mock-up) del Requirements analysis Document.

1.3 Documentazione dell’interfacce

Nell’implementazione del sistema i programmatori dovranno attenersi alle linee guida definite di seguito.

1.3.1 Classi e interfacce Java

Nella scrittura di codice per le classi Java ci si atterrà allo standard di Google relativo alla programmazione in Java consultabile al seguente link:

http://google.github.io/styleguide/javaguide.html#s4.6-whitespace

In questo standard ogni metodo ed ogni file possono non essere preceduti da un

commento, inoltre, potranno esserci commenti in merito a particolari decisioni o calcoli.

La convenzione utilizzata per i nomi delle variabili è la seguente:

*camelCase*.

(Aggiungere foto cammello:

https://en.m.wikipedia.org/wiki/Camel\_case)

Essa consiste nello scrivere parole composte in modo che ogni parola al centro della frase inizia con una lettera maiuscola.

Quando si codificano classi e interfacce Java, si dovrebbero rispettare le seguenti regole di formattazione:

1. Non si devono inserire spazi tra il nome del metodo e la parentesi tonda “(” che apre la lista dei parametri.

2. La parentesi graffa aperta “{“ si deve trovare sulla stessa linea dell’istruzione di dichiarazione.

3. La parentesi graffa chiusa “}” si troverà su una nuova riga vuota allo stesso livello di indentazione del nome della classe o dell’interfaccia.

I nomi delle classi devono essere sostantivi con l'iniziale maiuscola. I nomi delle classi dovrebbero essere semplici, descrittivi e inerenti al dominio applicativo. Non devono essere usati underscore per legare nomi.

I nomi dei metodi iniziano con una lettera minuscola, non sono consentiti caratteri speciali e dovranno essere semplici, descrittivi e inerenti al dominio applicativo.

Sia i nomi delle classi che i nomi dei metodi dovranno seguire la notazione camelCase descritta sopra.



1.3.2 Pagine lato Server (JSP)

Le pagine JSP quando eseguite dovranno produrre un documento conforme allo standard HTML 5.

Il codice Java presente nelle JSP deve aderire alle convenzioni descritte nel punti 1.3.1, con le seguenti specifiche:

1. Il tag di apertura (<%) è seguito immediatamente dalla fine della riga;

2. Il tag di chiusura (%>) si trova all'inizio di una riga;

3. È possibile evitare le due regole precedenti, se il corpo del codice Java consiste in una singola istruzione (<%= %>).



1.3.3 Pagine HTML

Le pagine HTML devono essere conformi allo standard HTML 5 e il codice deve essere identanto, per facilitare la lettura, secondo le seguenti regole:

1. Un'indentazione consiste in una tabulazione;

2. Ogni tag deve avere un'indentazione maggiore del tag che lo contiene;

3. Ogni tag di chiusura deve avere lo stesso livello di indentazione del corrispondente tag di apertura;

4. I tag di commento devono seguire le stesse regole che si applicano ai tag normali.

1.3.4 Script Javascript

Gli Script in Javascript devono rispettare le seguenti convenzioni:

1. Gli script che svolgono funzioni distinte dal rendering della pagina dovrebbero essere collocati in file dedicati.

2. Il codice Javascript deve seguire le stesse convenzioni per il layout e i nomi del codice Java.

3. Le funzioni Javascript devono essere documentate in modo analogo ai metodi Java.

1.3.5 Fogli di stile CSS

I fogli di stile (CSS) devono seguire la seguente convenzione:

Tutti gli stili non inline devono essere collocati in fogli di stile separati.

Inoltre ogni regola CSS deve essere formattata come segue:

1. I selettori della regola si devono trovare nella stessa riga;

2. L'ultimo selettore della regola è seguito da parentesi graffa aperta "{";

3. Le proprietà che costituiscono la regola sono listate una per riga e sono indentate rispetto ai selettori;

4. La regola è terminata da una parentesi graffa chiusa "}" collocata da sola su una riga;

1.3.6 Database SQL

I nomi delle tabelle devono seguire le seguenti regole:

1. Devono essere costituiti da sole lettere maiuscole;

2. Il nome deve essere un sostantivo singolare tratto dal dominio del problema ed esplicativo del contenuto.

I nomi dei campi delle tabelle devono seguire le seguenti regole:

1.1 Il nome deve rispettare la convenzione camelCase.

1.2 Il nome deve essere un sostantivo singolare tratto dal dominio del problema ed esplicativo del

contenuto.