BeHub

*Object Design Document*

# Introduzione

Dopo aver fatto il cristo…

# Linee Guida

Naming Conventions

I nomi dovranno rispettare le seguenti caratteristiche:

* Descrittivi
* Di uso comune
* Non abbreviati
* Utilizzando solo caratteri alfanumerici
* Cercare di mantenere una lunghezza medio-corta

Variabili

I nomi delle variabili dovranno iniziare con lettera minuscola e le parole successive dovranno avere la prima lettera maiuscola.

La dichiarazione delle variabili viene fatta di norma all’inizio facendo in modo che vi sia una sola variabile per riga, rendendo così più leggibile e chiaro il tutto.

Metodi

Per i nomi dei metodi vige la stessa regola delle variabili.

Nella maggior parte dei casi il nome del metodo sarà un verbo che rappresenta la sua funzione e sarà seguito dal nome dell’oggetto su cui opera.

Per quanto riguarda i vari metodi di accesso e modifica di una variabile dovranno sempre avere la seguente forma: getPassword(), setPassword();

Commenti

I commenti, se utilizzati, dovranno essere:

* Per variabili: scritti sulla stessa riga
* Per metodi: scritti appena dopo quest’ultimo è stato dichiarato

Classi ed Interfacce

I nomi delle classi e delle interfacce dovranno seguire la “CamelCase”; quindi, ogni parola inizia con la lettera maiuscola.

I nomi delle classi dovranno essere, come per i metodi, evocativi.

La dichiarazione di una classe è caratterizzata da:

1. Dichiarazione della classe public

2. Dichiarazioni di costanti con possibile commento

3. Dichiarazioni di variabili di classe con possibile commento

4. Dichiarazioni di variabili d’istanza con possibile commento

5. Costruttore

6. Possibile commento e dichiarazione metodi e variabili

Pagine

I nomi delle pagine dovranno essere scritti in minuscolo e vi sarà un “ – “ a dividere le parole.

Acronimi ed abbreviazioni

Acronimi:

● RAD: Requirements Analysis Document

● SDD: System Design Document

● ODD: Object Design Document

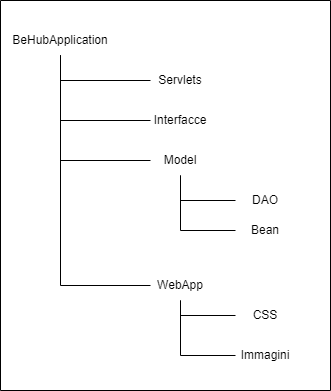
Abbreviazioni:

● DB: Database

Design Pattern

Data Access Object (DAO), è un pattern…

# Packages:



Il nostro sistema è diviso in tre livelli (architettura three-tier MVC)

• Presentation layer

• Application layer

• Storage layer

Il package BeHub contiene sottopackage che a loro volta contengono classi dedicate.

• Presentation layer:

Rappresenta la parte del sistema che si occupa dell’interfaccia utente, contiene quindi le risorse atte ad interagire con l’utente; contiene anche i form di input/output e le servlet.

• Application layer

Contiene DAO e Bean, cioè le risorse atte ad interfacciarsi con il database e a modellare i dati.

• Storage layer

Consiste in un DBMS che ha il compito di gestire il database per memorizzare, aggiornare e cancellare i dati dell’applicazione nonché di restituirli e ricevere le richieste dall’application layer.

Il *Package Servlets* contiene al suo interno tutte le servlet che rappresentano i servizi offerti e si occuperanno dell’elaborazione dei dati;

Il *Package Model* contiene tutti i Bean e tutti i DAO, i quali rappresentano i servizi offerti dai sottosistemi.

Il *Package Interfacce* contiene al suo interno tutte le interfacce che saranno implementate rispettivi dao e dal gestore mail.

Il *Package WebApp* contiene tutte le componenti che raffigurino la presentazione del sistema, quindi contiene HTML, JSP e CSS.

La suddivisione dei package in servlets, model e webapp è motivata, come detto all’inizio, dall’utilizzo dell’architettura MVC, che ci permette di separare la logica di business (Servlets) dalla presentazione (WebApp) e dalla gestione dei dati (Model). Il package model è a sua volta suddiviso in bean e dao, come anche il package webapp che è suddiviso in css ed immagini.