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 prima 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. Costruttore

3. Possibile commento e dichiarazione metodi e variabili

4. Dichiarazioni di costanti con possibile commento

5. Dichiarazioni di variabili di classe con possibile commento

6. Dichiarazioni di variabili d’istanza con possibile commento

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

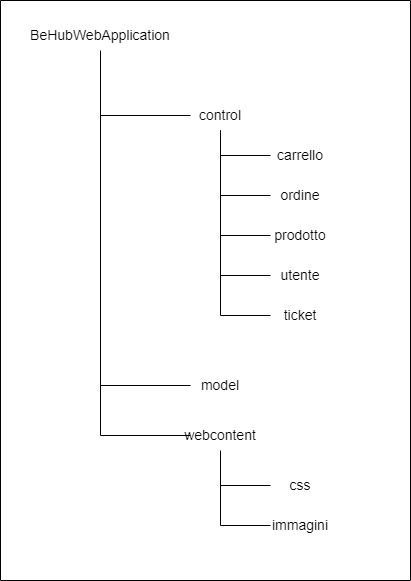
**MVC Pattern**

Lo schema che abbiamo identificato è esattamente quello proposto dal pattern MVC. In particolare:

* *Model:* contiene i metodi di accesso ai dati.
* *View:* si occupa di visualizzare i dati all'utente e gestisce l'interazione fra quest'ultimo e l'infrastruttura sottostante(webcontent).
* *Controller:* riceve i comandi dell'utente attraverso il View e reagisce eseguendo delle operazioni che possono interessare il Model e che portano generalmente ad un cambiamento di stato del View.

La suddivisione dei package in servlets, model e webcontent è per l’appunto motivata dall’utilizzo dell’architettura MVC, la quale permette di separare la logica di business (servlets) dalla presentazione (webcontent) e dalla gestione dei dati (model). Il package model è a sua volta suddiviso in bean e dao. Le servlet nel package servlets si occuperanno del

# Packages:



I tre pacchetti principali sono control, model e webcontent.

Il pacchetto control è suddiviso a sua volta in altri sotto pacchetti che presentato al proprio interno le servlet.

Nel pacchetto model abbiamo, non solo i model, ma sono presenti anche tutti gli oggetti entity che definiscono gli oggetti di dominio e i quali saranno utilizzati come bean dalle servlet presenti nel package control.

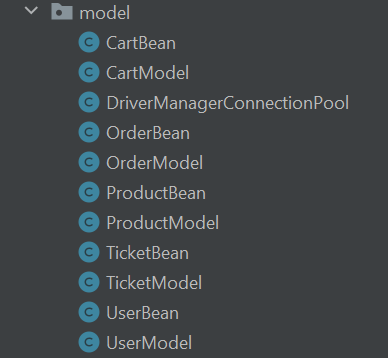
Infine, il pacchetto webcontent oltre ad avere tutti i file jsp, sorgenti HTML e js presenta altri due pacchetti: uno contenente tutti i file css e l’altro tutte le immagini usate nel sito.

Control Package

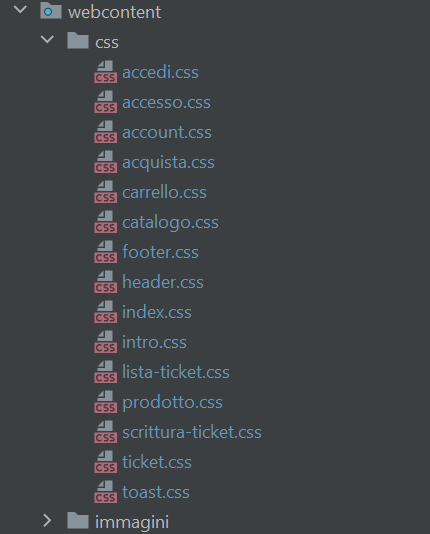
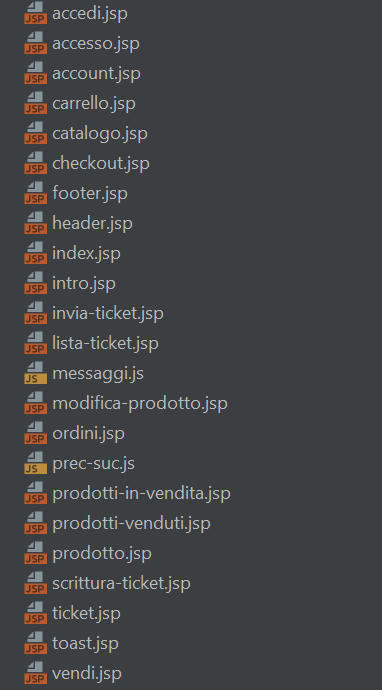
Immagine che contiene testo

Descrizione generata automaticamente

Model Package



WebApp Package



# Interfacce delle Classi