Università degli Studi di Salerno

Corso di Ingegneria del Software

iDea PC

Sito e-commerce dedicato alla vendita di componenti pc

Versione 2.0

Data 25/01/2018

Partecipanti:

Nome	Matricola
Aquilino Leone	0512102290
Domenico Capasso	0512102272

Scritto da:	Leone Aquilino, Domenico Capasso

Revision History

Data	Versione	Descrizione	Autore
10/01/2018	0.1	Prima Stesura	Leone Aquilino, Domenico Capasso
11/01//2018	1.0	Completamento ODD	Leone Aquilino, Domenico Capasso
15/01/2018	1.1	Correzioni varie	Leone Aquilino, Domenico Capasso
25/01/2018	2.0	Completamento ODD seconda stesura	Leone Aquilino, Domenico Capasso

INDICE

1 Introduzione

Questo documento descrive gli obbiettivi del design imposti dagli sviluppatori e le linee guida per lo sviluppo delle interfacce dei sottosistemi e la suddivisione dei sottosistemi in package e classi.

1.1 Object Design trade-offs

Quando si definiscono questi obiettivi, spesso solo un piccolo sottoinsieme di essi può essere tenuto in considerazione. Perciò, si deve dare delle priorità agli obiettivi di design, tenendo conto anche di aspetti manageriali, quali il rispetto del budget. In genere, quindi, si utilizzano dei trade-off tra i principali obiettivi di design.

- **Comprensibilità vs costi**

La comprensibilità del codice è un aspetto molto importante soprattutto per la fase di testing. Ogni classe e metodo deve essere facilmente interpretabile anche da chi non ha collaborato al progetto. Nel codice si useranno i commenti standard e Javadoc per aumentare la comprensione del codice sorgente. Ovviamente questa caratteristica aggiungerà dei costi allo sviluppo del nostro progetto.

- **Prestazioni vs Costi**

Non avendo a disposizione alcun budget, utilizziamo materiale open source per la realizzazione del software IDeaPC, tuttavia esso andrà ad impattare sulle prestazioni del sistema perché l'avvio del server, da parte di Tomcat, non permetterà subito di visualizzare i contenuti della piattaforma; perciò, verranno assicurati 10 secondi entro il quale il sistema risponderà alle azioni dell'utente.

- **Costi vs Mantenimento**

L'utilizzo di materiale open source e del linguaggio Javadoc rende il sistema facilmente comprensibile e modificabile, ma il mantenimento sarà difficile da gestire perché, quanti più dati immetteremo nel nostro database, tanto più l'open source usato per il database avrà problemi a effettuare risposte alle query e, quindi, i ritardi aumenteranno.

- **Interfaccia vs Easy-use**

L'interfaccia grazie all'utilizzo delle form si presenta semplice ed intuitiva, permettendo una facile gestione del database anche ai meno esperti col computer. (Easy-Use)

- **Prestazioni vs Sicurezza**

Il sistema prevede l'autenticazione in modo da garantire l'accesso di persone autorizzate e la privacy dei propri dati; tuttavia tutto questo va a discapito delle prestazioni, che saranno più lente in virtù della chiamata al database per controllare le credenziali inserite.

1.2 Linee guida per la documentazione delle interfacce

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

1.2.1 Classi Java

Ogni file sorgente Java contiene una singola classe pubblica.

I file sorgente devono essere strutturati:

- Istruzione package.
- Istruzioni import.

    - package interni;
    - package estensioni;
    - package libreria di terze parti;
    - package applicativi;

- Descrizione generale della classe. Questa sezione descrive in modo sintetico le caratteristiche del file. Il template è il seguente:

```
/*
* NomeClasse
* Breve descrizione della classe
*/
```

- Dichiarazione di classe.
- Descrizione dei metodi. Il template è il seguente:

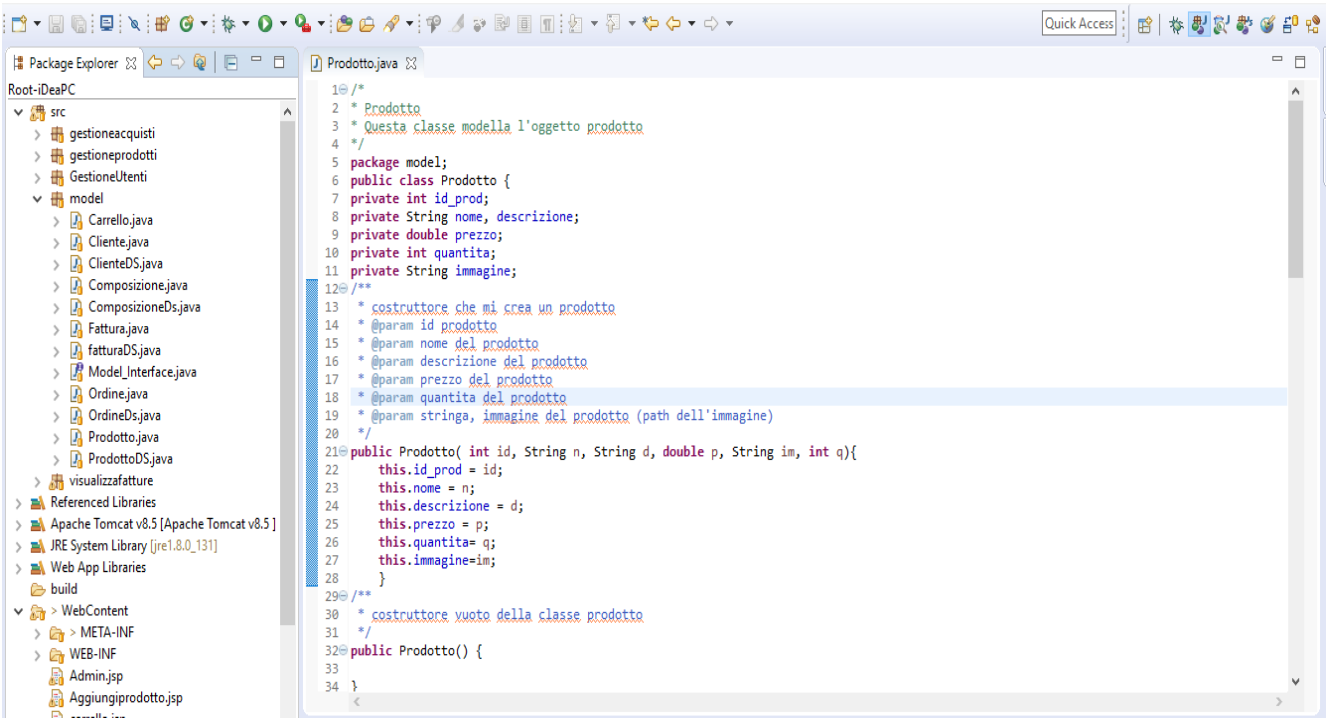
```
/**
* metodo che stampa un testo
* @param text testo da stampare
* @return true valore booleano
*/
public boolean metodo(String text)
{
        System.out.println(text);
        return true;
}
```

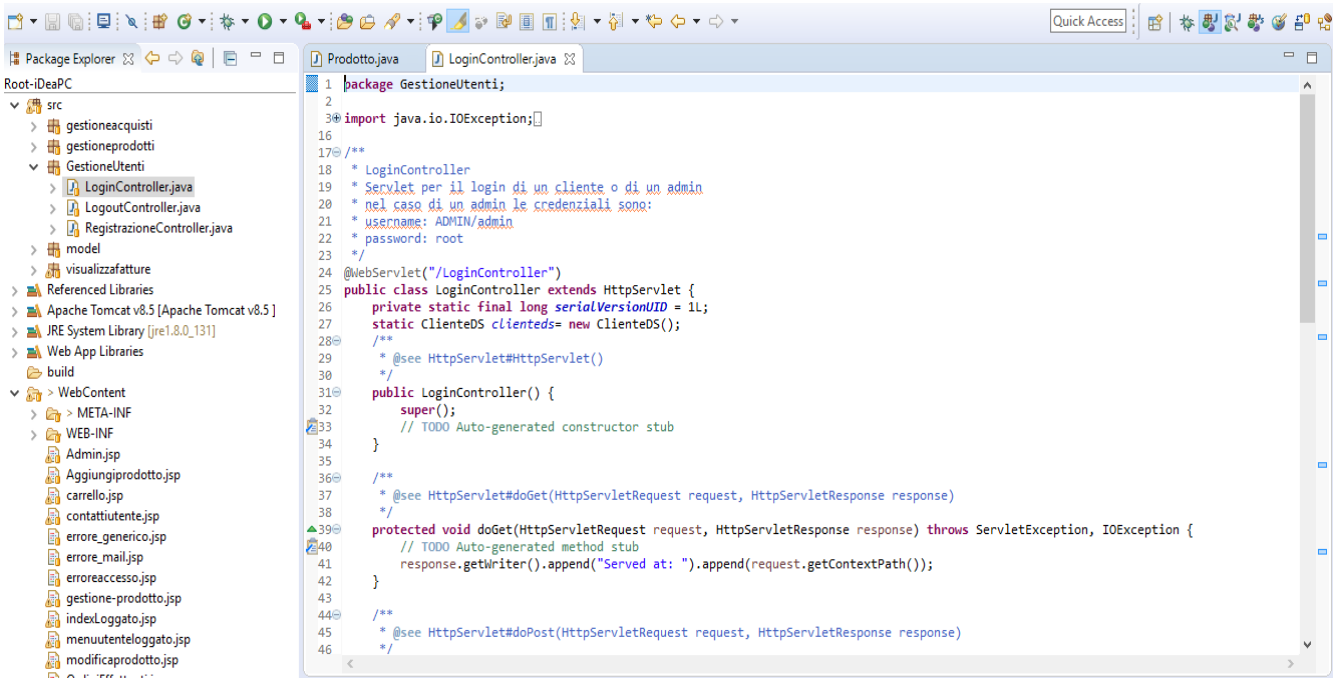
- Dichiarazioni di metodi.

La dichiarazione è formata da:

- Commento: opzionale.
- Istruzioni dichiarative della classe.
- Variabili d'istanza.
    - attributi pubblici;
    - attributi privati;
    - attributi protetti;
    - altri attributi;
    - costruttori;
    - altri metodi;
- Costruttori.
- Commento e dichiarazione metodo.

Esempio di classe:





1.2.1 Pagine Java lato Server (JSP)

Le pagine JSP devono aderire alle convenzioni per la codifica in Java, con le seguenti puntualizzazioni:

- I tag di apertura (<%) e chiusura (%>) si trovano da soli in una riga.
- Il contenuto dei tag si trova al livello successivo di indentazione.
- È possibile emendare alle due regole precedenti, se il corpo del codice Java consiste in una singola istruzione.

1.2.2 Script Javascript

Gli script che svolgono funzioni distinte dal mero rendering della pagina sono collocati all'interno delle pagine jsp. Il codice Javascript deve seguire le stesse convenzioni per il layout e i nomi del codice Java. I documenti Javascript devono essere iniziati da un commento analogo a quello presente nei file Java. Le funzioni Javascript devono essere documentate in modo analogo ai metodi Java. Gli oggetti Javascript devono essere preceduti da un commento in stile Javadoc, che segue il seguente formato:

```
/**
* Descrizione breve
* Eventuale ulteriore descrizione
* Specifica degli argomenti del costruttore (@param)
*
* Metodo nomeMetodo1
* Descrizione breve
* Eventuale ulteriore descrizione
* Specifica degli argomenti (@param)
* Specifica dei risultati (@return)
*
```

```
* Metodo nomeMetodo2
* Descrizione breve
* Eventuale ulteriore descrizione
* Specifica degli argomenti (@param)
* Specifica dei risultati (@return)
*
* ...
*/
function EseguiX(a, b, c) {
```

1.2.3 Fogli di stile CSS

Tutti gli stili non inline devono essere collocati in fogli di stile separati, mentre per quelli inline devono essere posizionati all'interno del tag. Ogni regola CSS deve essere formattata come segue:

- I selettori della regola si trovano a livello 0 di indentazione, uno per riga;
- L'ultimo selettore della regola è seguito da parentesi graffa aperta ({);
- Le proprietà che costituiscono la regola sono listate una per riga e sono indentate rispetto ai selettori;
- La regola è terminata da una parentesi graffa chiusa (}), collocata da sola su una riga;

Le proprietà e le regole poco chiare dovrebbero essere precedute da un commento esplicativo. Le regole possono essere divise in blocchi concettualmente legati, preceduti da commenti in stile Javadoc, che ne spiegano lo scopo.