Università degli Studi di Salerno
Corso di Ingegneria del Software

NowIWant
Object Design Document
Versione 1.0

LOGO PROGETTO

Data: 20/06/2018

Progetto: NowIWant	Versione: 1.0
Documento: object design document	Data: 20/06/2018

Coordinatore del progetto:

Nome	Matricola
NA	NA

Partecipanti:

Nome	Matricola
Ferrazzano Pompeo Alessio	0512102928
Citro Antonio	0512102922
Giovanni Lembo	0512103252
Robertazzi Gennaro Alessio	0512103792

Scritto da:	FPA, CA, LG, RGA

Revision History

Data	Versione	Descrizione	Autore
16/12/2017	0.1	Introduzione ODD	FPA, RGA
17/12/2017	0.5	Packaging e class interfaces	CA, LG
18/12/2017	1.0	Documentazione del riuso, Glassario e revisione	FPA, CA, LG, RGA

Indice

1. Introduzione

1.1 Object Design Trade-off

Dopo la finalizzazione del documento RAD (Requirement Analysis Document) e l'SDD (System Design Document), abbiamo descritto quello che sarà il nostro sistema e dunque i nostri obiettivi, tralasciando gli aspetti di implementazione. Per quanto riguarda la realizzazione del sistema NowIWant, sono stati individuati i seguenti trade-off.

1.1.1 Prestazioni vs Robustezza

Per favorire un aumento della robustezza, abbiamo deciso di inserire dei controlli sia sul client che sul server. Chiaramente, ciò ha comportato, una leggera diminuzione delle prestazioni.

1.1.2 Comprensibilità vs Tempo

Costituita da codice comprensibile anche per eventuali persone esterne del progetto, prevede l'utilizzo di classi con metodi ben identificabili e facilmente interpretabili, utilizzando l'indentazione ed una documentazione appropriata del codice sorgente. Questo vantaggio comporta, però un incremento del tempo per lo sviluppo e realizzazione del sistema.

1.1.3 Sicurezza vs Performance

Per incrementare il livello di sicurezza abbiamo deciso di mantenere la sessione sul server. Chiaramente, ciò ha comportato, una leggera diminuzione delle performance.

1.1.4 Costi vs mantenimento

Si è deciso di sviluppare un pannello di controllo per la gestione dei dati persistenti. Ciò ha comportato un aumento dei costi dell'intero progetto.

La scelta di utilizzare un database è scaturita dai diversi vantaggi che se ne derivano:

- Gestione consistente dei dati;
- Tempo risposta basso rispetto all'utilizzo di un file system;
- Accesso veloce e concorrente ai dati.

Ovviamente l'utilizzo di un database comporta l'utilizzo di una grande quantità di memoria nel momento il cui la mole dei dati dovesse essere notevole.

1.2 Linee guida per la documentazione delle interfacce

1.2.1 Classi e interfacce Java

Nella scrittura di codice per le classi Java ci si atterrà allo standard [CCJPL] nella sua interezza, con particolare attenzione a:

1. Convenzioni sui nomi (cap. 9 dello standard);
2. Struttura dei file (cap. 3 dello standard);
3. Accesso alle variabili (sez. 10.1 dello standard);
4. Commenti speciali (sez. 10.5.4 dello standard);
5. Utilizzo degli spazi bianchi (cap. 8 dello standard);

Si praticano, inoltre, le seguenti restrizioni e variazioni:

1. All'inizio di ogni file devono essere presenti alcune informazioni strutturate come segue:

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

2. Tutte le variabili dovrebbero essere private (private).
3. Tutte le variabili che non vengono mai modificate dovrebbero essere dichiarate come costanti (final).
4. Si devono usare le tabulazioni per gestire l'indentazione.

1.2.2 Base di dati

Le tabelle della base di dati dovrebbero rispettare la terza forma normale di Codd (3NF). Ove ciò
non si verifichi, tale fatto deve essere riportato e motivato nella documentazione della base di dati.
Le scelte di gestione nel trattamento dell'integrità referenziale devono essere riportate e motivate nella documentazione della base di dati.

1.2.3 Pagine lato Server (JSP)

Le pagine JSP devono, quando eseguite, produrre, in ogni circostanza, un documento conforme allo standard HTML versione 5.
Le parti Java delle pagine devono aderire alle convenzioni per la codifica in Java, con le seguenti puntualizzazioni:

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 emendare alle due regole precedenti, se il corpo del codice Java consiste in una singola istruzione:

```
<!-- Accettabile -->
<% for (String par : paragraphs) {%>
<p class='item'><% out.print(par); %></p>
<% } %>


<!-- Non accettabile -->
<p class='item'><% List<String> paragraphs = getParagraphs();
out.print(paragraphs.get(i++));%></p>
```

1.2.4 Pagine HTML

Le tabelle della base di dati dovrebbero rispettare la terza forma normale di Codd (3NF). Ove ciò
non si verifichi, tale fatto deve essere riportato e motivato nella documentazione della base di dati.
Le scelte di gestione nel trattamento dell'integrità referenziale devono essere riportate e motivate
nella documentazione della base di dati. Le pagine HTML, statiche e dinamiche, devono essere totalmente aderenti allo standard HTML versione 5.
Inoltre, il codice HTML statico deve utilizzare l'indentazione, 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;

```
<!-- Accettabile -->
<div><span><nl>
<li>Uno</li>
<li>
Due
</li>
</nl></span></div>

<!-- Non accettabile -->
<div><span>
<nl>
<li>Uno</li>
<li>
Due
</li>
</nl></span>
</div>
```

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

1.2.5 Script Javascript

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

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 ClasseX(a, b, c) {
```

1.2.6 Fogli di stile CSS

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

Ogni foglio di stile deve essere iniziato da un commento analogo a quello presente nei file Java.

Ogni regola CSS deve essere formattata come segue:

1. I selettori della regola si trovano a livello 0 di indentazione, uno per 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;

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, e seguiti da 2 righe bianche.

1.2.7 Database SQL

Le tabelle della base di dati dovrebbero rispettare la terza forma normale di Codd (3NF). Ove ciò non si verifichi, tale fatto deve essere riportato e motivato nella documentazione della base di dati.

Le scelte di gestione nel trattamento dell'integrità referenziale devono essere riportate e motivate nella documentazione della base di dati. Le tabelle del database dovrebbero essere in terza forma normale di Codd (3NF). Qualora non lo fossero, la cosa deve essere documentata e motivata nello script di creazione del database.

I nomi delle tabelle devono seguire le seguenti regole:

- devono essere costituiti da sole lettere maiuscole;
- il nome deve essere un sostantivo singolare tratto dal dominio del problema ed esplicativo del contenuto.

I nomi dei campi devono seguire le seguenti regole:

- devono essere costituiti da sole lettere minuscole;
- se il nome è costituito da più parole, queste sono separate da un underscore (_);
- il nome deve essere un sostantivo singolare tratto dal dominio del problema ed esplicativo del contenuto.

1.3 Definizioni, acronimi ed abbreviazioni

CCJPL: Code Conventions for the Java Programming
RAD: Requirements Analysis Document
SDD: System Design Document
ODD: Object Design Document
SQL: Structured Query Language
DB: DataBase
DBMS: DataBase Management System
JSP: Java Server Pages
HTML: HyperText Markup Language
CSS: Cascading Style Sheets
3NF: Terza Froma Normale
API: Application Programming Interface
BROWSER: Chrome-Firefox-IE
WebBrowser: Cliente-Amministratore (Utente che accede al sistema)
WebServer: Server gestore dei Database

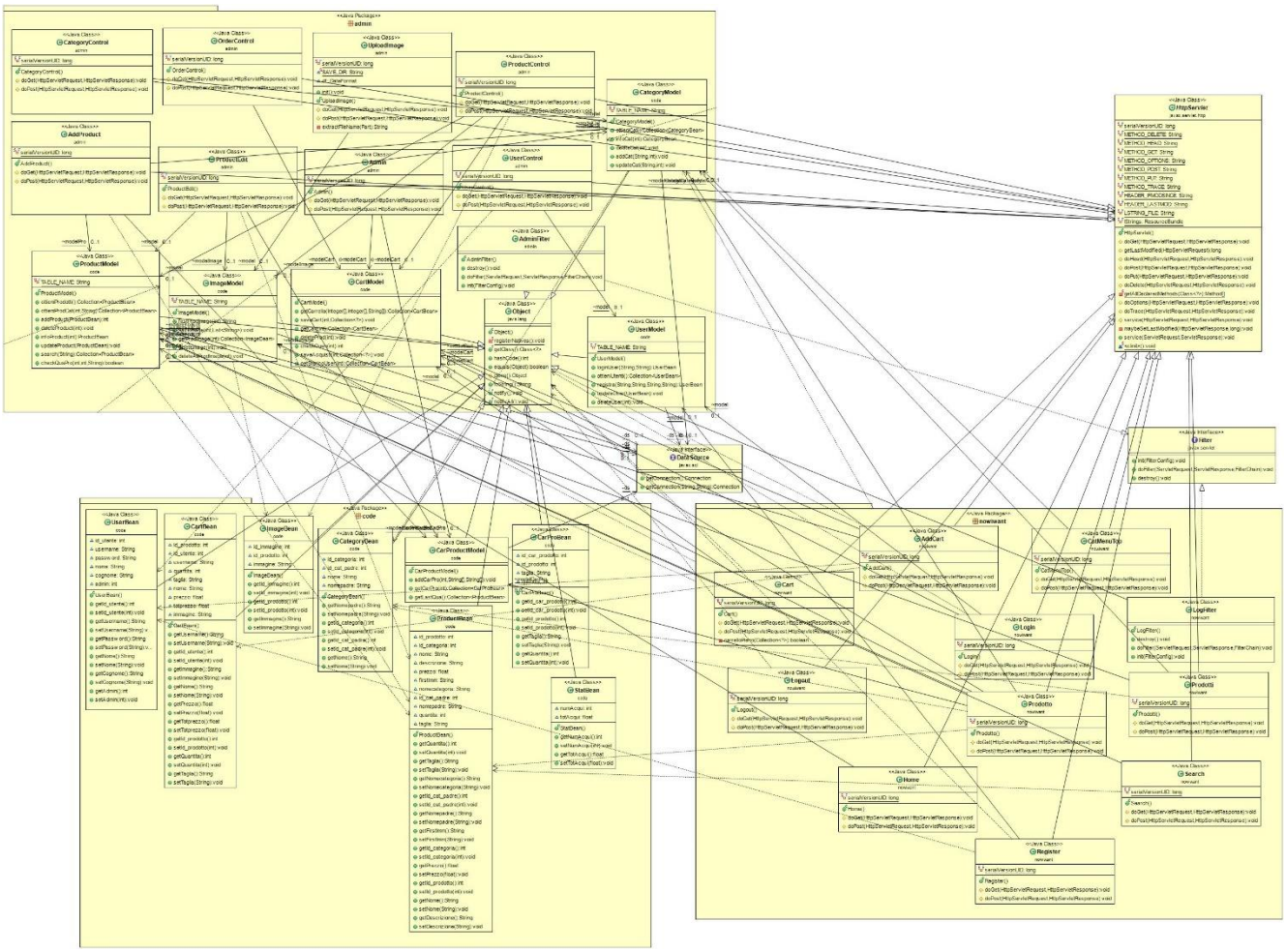
1.4 Riferimenti

Il presente ODD riferisce alle ultime versioni dei precedenti documenti rilasciati:

- RAD_NOWIWANT;
- SDD_NOWIWANT.

2. Packaging e class interfaces

Il packing e class intefaces è stato realizzato tramite l'applicativo javadoc utilizzato per la generazione automatica del codice sorgente scritto in linguaggio java.



DBMANAGER

NOWIWANT

-URL: String
-CONNESSIONE: String
-st: Statement
-con: Connection

utenti(id_utente,username,password,nome,cognome,admin)
prodotti(id_prodotto,id_categoria,nome,descrizione,prezzo,date_add)
immagini(id_immagine,id_prodotto,immagine)
categorie(id_categoria,id_cat_padre,nome)
car_prodotti(id_car_prodotto,id_prodotto,taglia,quantita)
carrelli(id_carrello,id_utente,id_prodotto,quantita,taglia)
acquisti(id_acquisto,id_utente,id_prodotto,taglia,prezzo,quantita,date_add)

3. Documentazione del riuso

3.1 Design Pattern

Singleton:

Un degli aspetti critici del funzionamento del sistema è l'accesso ai dati persistenti. Per questo motivo si è scelto di delegare ad una singola classe la responsabilità di gestire le connessioni al database. Il problema è la possibilità di ottenere più istanze della classe. Per evitare questo problema si è deciso di applicare nella classe il design pattern Singleton.

4. Glossario

Account: e-mail e password dell'utente registrato; obbligatorie ai fini della registrazione.

Prodotto: descrizione, corredata da immagine, di un prodotto che un utente della piattaforma ha messo in vendita.

DBMS: Database Management System, software progettato per la creazione e la manipolazione di database.

Design Pattern: Un design pattern (traducibile in lingua italiana come schema progettuale, schema di progettazione, schema architetturale), è un concetto che può essere definito "una soluzione progettuale generale ad un problema ricorrente". Si tratta di una descrizione o modello logico da applicare per la risoluzione di un problema che può presentarsi in diverse situazioni durante le fasi di progettazione e sviluppo del software.

HTML: L'Hyper Text Markup Language (HTML; traduzione letterale: linguaggio a marcatori per ipertesti), in informatica è il linguaggio di markup solitamente usato per la formattazione e impaginazione di documenti ipertestuali disponibili nel World Wide Web sotto forma di pagine web.

Javadoc: Javadoc è un applicativo incluso nel Java Development Kit della Sun Microsystems, utilizzato per la generazione automatica della documentazione del codice sorgente scritto in linguaggio Java.

ODD: Object Design Document.

Profilo: insieme delle informazioni (residenza, foto profilo, descrizione, contatti) che un utente può aggiungere, facoltativamente, in seguito alla registrazione.

RAD: Requirements Analysis Document, documento che tratta nel dettaglio l'analisi dei requisiti.

SDD: System Design Document, documento che tratta nel dettaglio della progettazione del sistema e dei suoi obiettivi.