

**INGEGNERIA DEL SOFTWARE 2016-2017**

**Documento di Object Design 0.1**

**Fiorazon**

**15/12/2016**

**Sommario**

[**1.** **Introduction** 3](#_Toc469562348)

[**I.** **Object design trade-offs** 3](#_Toc469562349)

[**II.** **Interface documentation guidelines** 4](#_Toc469562350)

[**III.** **Definitions, acronyms, and abbreviations** 4](#_Toc469562351)

[**IV.** **References** 5](#_Toc469562352)

[**2.** **Packages** 5](#_Toc469562353)

[**3.** **Class Interfaces** 6](#_Toc469562354)

[**4.** **Team** 8](#_Toc469562355)

[**5.** **Storia delle versioni** 8](#_Toc469562356)

### **Introduction**

### **Object design trade-offs**

La nostra piattaforma sarà completamente sviluppata senza l’ausilio di codice scritto da terzi.

I motivi di questa scelta ricadono sul fatto che il riutilizzo di codice di terzi:

* Appesantirebbe il caricamento delle pagine web e aumenterebbe il tempo di computazione dei servizi offerti dal server, in quanto il codice avrà molto probabilmente funzioni inutilizzate nel normale utilizzo dell’applicazione stessa;
* Utilizzare codice preconfezionato da terzi produce l’inconveniente di doverlo modificare in caso di un bug o di una modifica su qualche funzioni, il che non sarebbe tanto semplice e veloce come soluzione considerando che il codice non è stato prodotto da nessun membro del progetto.

Nell’ applicazione esistono alcune situazioni che potrebbero provocare errori nel sistema e vengono gestiti nei seguenti modi:

* **Username o password sbagliata:**

riscontrando questo problema il sistema mostra un messaggio d’ errore che identifica il problema.

* **Inserimento data nascita che identifica un utente minorenne nella fase di registrazione oppure nella fase della modifica dell’anagrafica dell’utente:**

Il sistema riscontrando il problema in primo luogo impedisce la modifica o la registrazione e inoltre nella pagina in cui si trova a seconda del caso compare un messaggio di errore che lo identifica.

* **Utilizzo di un username già utilizzata da un altro utente nella fase di registrazione:**

L’utilizzo di un username già utilizzata non è permesso, quindi di conseguenza il sistema non permette di continuare la registrazione e quindi mostra un messaggio di errore che ne spiega i motivi e mostra alcuni username no utilizzate che potrebbero andare bene.

* **Inserimento di un prodotto già esistente nel catalogo:**

Per evitare problemi di incongruenza nei dati dei prodotti è fatto assoluto divieto inserire un prodotto già esistente. Il sistema riscontrando il problema non permette di continuare l’operazione e mostra un messaggio di errore nella pagina.

* **Procedere con l’ordine del carrello senza essere loggati:**

L’ applicazione per permettere di effettuare un ordine obbliga l’utente ad effettuare il login oppure la registrazione se non l’ha fatto ancora. Quindi il sistema trovandosi nel punto in cui un utente non registrato voglia produrre un ordine, gli mostra un massaggio di errore e contemporaneamente anche la pagina che permetta di fare il login oppure la registrazione.

### **Interface documentation guidelines**

|  |  |  |
| --- | --- | --- |
| **Identifier Type** | **Rules for Naming** | **Examples** |
| **Package** | Il nome di ogni package deve essere scritto tutto in minuscolo, inoltre deve essere un nome molto sintetico e al contempo deve essere rispecchiare il suo contenuto | package beans |
| **Classe** | Il nome di ogni classe deve iniziare per lettera maiuscola, mentre le restanti lettere dovranno essere minuscole ad eccezione di un nome composto da più parole, in cui e consentito differenziare la parola mettendo la prima lettera di essa in maiuscolo. E infine il nome deve essere fortemente significativo in modo da facilitare la comprensione del ruolo della suddetta classe. | Class GestisciCarrello |
| **Interfaccia** | I nomi delle interfacce seguono le stesse regole delle classi. | Interface ModificaCampo |
| **Metodo** | I nomi dei metodi devono iniziare per un verbo scritto in minuscolo, che rappresenti l’effetto del suo utilizzo.  Esso può essere succeduto da altre parole, le quali devono iniziare con lettera maiuscola e continuare fino alla fine in minuscolo. | inserisciProdotto(); |
| **Variabile** | I nomi delle variabili devono essere brevi e coincisi, e anch’esse devono iniziare per lettera minuscola, mentre le parole interne per maiuscola | Int contaUtenti; |
| **Costante** | I nomi delle costanti devono essere scritti tutti in maiuscolo e separati da “\_” se composti da più parole. | COSTANTE\_PI\_GRECO |

### **Definitions, acronyms, and abbreviations**

**Package:** collezione di classi e interfacce correlate.

**Classe:** è identificabile come un tipo di dato astratto.

**Interfaccia:** è un tipo analogo alla classe, ma vincolato a non definire l'implementazione dei propri metodi.

**Metodo:** è un sottoprogramma associato in modo esclusivo ad una classe e che rappresenta.

**Variabile:** è un insieme di dati modificabili situati in una porzione di memoria destinata a contenere dei dati **.**

**Costante:** identifica una porzione di memoria il cui valore non varia nel corso dell'esecuzione di un programma.

### **References**

* Requirements Analysis Document
* System Design Document

### **Packages**

Individuiamo 4 packages in cui dividere il codice, uno per ogni sotto-sistema:   
-carrelloPackage  
-prodottiPackage  
-utentiPackage  
-acquistoPackage

Il package carrelloPackage contiene i file “Carrello.jsp”, “ControlloCarrello.java”, “CarrelloManager.java”.   
Esso contiene tutte quelle pagine .jsp e .java responsabili dell’interazione con il carrello.  
Interagisce con l’acquistoPackage dal file “ControlloCarrello.java” verso “Pagamento.jsp” (Operazione di acquisto del carrello) e con prodottiPackage tramite “ControlloCarrello.java” chiamato da “Catalogo.jsp” (Operazione di aggiunta prodotto al carrello).

Il package prodottiPackage contiene i file “AggiungiProdotto.jsp”, “ControlloProdotti.java”, “ProdottiManager.java”, “Prodotto.java”, “Catalogo.jsp”, “CancellaProdotto.jsp”, “ModificaProdotto.jsp”, “Risultati.jsp”.   
Esso contiene tutte quelle pagine .jsp e .java responsabili delle operazioni riguardanti i prodotti.  
Interagisce con carrelloPackage tramite “Catalogo.jsp” verso “ControlloCarrello.java” (Operazione di aggiunta prodotto al carrello). Interagisce con utentiPackage tramite “AggiungiProdotto.jsp” da “PaginaAmministratore.jsp” (Operazione di aggiunta di un nuovo prodotto) e tramite “Catalogo.jsp” da “PaginaAmministratore.jsp”(Operazioni di cancellazione di un prodotto e di modifica di un prodotto).

Il package utentiPackage contiene i file “PaginaAmministratore.jsp”, “PaginaUtente.jsp”, “ControlloUtenti.java”, “UtentiManager.java”, “PaginaUtente.jsp”, “PaginaLogin.html”, “PaginaRegistrazione.jsp”.  
Esso contiene tutte quelle pagine .jsp e .java responsabili di quelle operazioni riguardanti gli utenti e l’amministratore.  
Interagisce con acquistoPackage tramite “PaginaAmministratore.jsp” verso “OrdiniAmministratore.jsp” (Operazione di avanzamento dello stato di spedizione di un ordine), tramite “PaginaUtente.jsp” verso “OrdiniAmministratore.jsp” (Operazione di Conferma dell’arrivo di un ordine). Interagisce con prodottoPackage tramite “PaginaAmministratore.jsp” verso “AggiungiProdotto.jsp” (Operazione di aggiunta di un nuovo prodotto), verso “Catalogo.jsp” (Operazione di cancellazione di un prodotto e di modifica di un prodotto).

Il package acquistoPackage contiene i file “OrdiniAmministratore.jsp”, “ControlloOrdini.java”, “Ordine.java”, “OrdiniManager.java”, “Pagamento.jsp”.   
Esso contiene quei file responsabili di eseguire l’acquisto e di gestire la vita di un ordine.  
Interagisce con utentiPackage tramte “OrdiniAmministratore.jsp” da “PaginaAmministratore.jsp” (Operazione di avanzamento dello stato di spedizione di un ordine) e da “PaginaUtente.jsp” (Operazione di Conferma dell’arrivo di un ordine). Interagisce con carrelloPackage tramite “Pagamento.jsp” da “ControlloCarrello.java” (operazione di acquisto del carrello).

### **Class Interfaces**

Nel package carrelloPackage si individuano 2 classi:

ControlloCarrello.java  
Questa classe è il control object responsabile delle operazioni di gestione del carrello

+controlloAcquistoCarrello()   
Questo metodo permette di acquistare il carrello

+aggiungiProdottoCarrello()   
Questo metodo permette di aggiungere un prodotto al carrello

+rimuoviProdottoCarrello()   
Questo metodo permette di rimuovere un prodotto dal carrello

CarrelloManager.java  
Questa classe è il model object responsabile delle operazioni di accesso al database della classe ControlloCarrello.java effettuando le operazioni di persistenza dei dati relative al carrello.

+controlloAcquistoCarrello()  
Questo metodo permette di acquistare il carrello

+aggiungiProdottoCarrello()  
Questo metodo permette di aggiungere un prodotto al carrello

+rimuoviProdottoCarrello()  
Questo metodo permette di rimuovere un prodotto dal carrello

Nel package prodottiPackage si individuano 2 classi:

ControlloProdotti.java  
Questa classe è il control object responsabile delle operazioni sui prodotti

+aggiungiProdotto()   
Questo metodo permette all’amministratore di aggiungere un nuovo prodotto

+caricaProdotti()   
Questo metodo permette di ricercare tutti i prodotti  
+cancellaProdotto()   
Questo metodo permette all’amministratore di eliminare un prodotto esistente

+modificaProdotto()   
Questo metodo permette all’amministratore di modificare un prodotto esistente

+ricercaProdotti()   
Questo metodo permette di effettuare una ricerca sui prodotti

ProdottiManager.java  
Questa classe è il model object responsabile delle operazioni di accesso al database della classe ControlloProdotti.java effettuando le operazioni di persistenza dei dati relative ai prodotti

+aggiungiProdotto()   
Questo metodo permette all’amministratore di aggiungere un nuovo prodotto

+caricaProdotti()   
Questo metodo permette di ricercare tutti i prodotti  
+cancellaProdotto()   
Questo metodo permette all’amministratore di eliminare un prodotto esistente

+modificaProdotto()   
Questo metodo permette all’amministratore di modificare un prodotto esistente

+ricercaProdotti()   
Questo metodo permette di effettuare una ricerca sui prodotti

Nel package utentiPackage si individuano 2 classi:

ControlloUtenti.java   
Questa classe è il control object responsabile delle operazioni sugli utenti

+controlloCancella()   
Questo metodo permette ad un utente di cancellarsi

+controlloLoginAmministratore()  
Questo metodo permette l’autenticazione dell’amministratore

+controlloLoginUtente()   
Questo metodo permette l’autenticazione dell’utente

+controlloLogoutUtente()   
Questo metodo permette il logout dell’utente

+controlloLogoutAmministratore()   
Questo metodo permette il logout dell’amministratore

+controlloModificaDati()   
Questo metodo permette ad un utente di modificare i propri dati personali

+controlloRegistrazione()   
Questo metodo permette ad un nuovo utente di registrarsi

UtentiManager.java   
Questa classe è il model object responsabile delle operazioni di accesso al database della classe ControlloUtenti.java effettuando le operazioni di persistenza dei dati relative agli utenti

+controlloCancella ()   
Questo metodo permette ad un utente di cancellarsi

+controllologinAmministratore()   
Questo metodo permette l’autenticazione dell’amministratore

+controlloLoginUtente ()   
Questo metodo permette l’autenticazione dell’utente

+controlloModificaDati()  
Questo metodo permette ad un utente di modificare i propri dati personali

+controlloRegistrazione()  
Questo metodo permette ad un nuovo utente di registrarsi

Nel package acquistoPackage si individuano 2 classi:

ControlloAcquisto.java   
Questa classe è il control object responsabile delle operazioni sugli ordini

+caricamentoOrdini()   
Questo metodo permette all’amministratore di ricercare tutti gli ordini

+controlloSpedizione()   
Questo metodo permette di far avanzare lo stato di spedizione di un ordine

+controlloPagamento()   
Questo metodo permette di effettuare l’acquisto del carrello

AcquistoManager.java   
Questa classe è il model object responsabile delle operazioni di accesso al database della classe ControlloAcquisto.java effettuando le operazioni di persistenza dei dati relative agli ordini

+caricamentoOrdini()  
Questo metodo permette all’amministratore di ricercare tutti gli ordini

+controlloSpedizione()  
Questo metodo permette di far avanzare lo stato di spedizione di un ordine

+controlloPagamento()  
Questo metodo permette di effettuare l’acquisto del carrello

### **Team**

|  |  |
| --- | --- |
| **Nome** | **Matricola** |
| Giuseppe Siani | 0512102958 |
| Carmine D’ Alessandro | 0512103402 |
| Aleandro Giuseppe Libano | 0512102976 |

### **Storia delle versioni**

|  |  |  |  |
| --- | --- | --- | --- |
| DATA | VERSIONE | DESCRIZIONE | AUTORE |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |