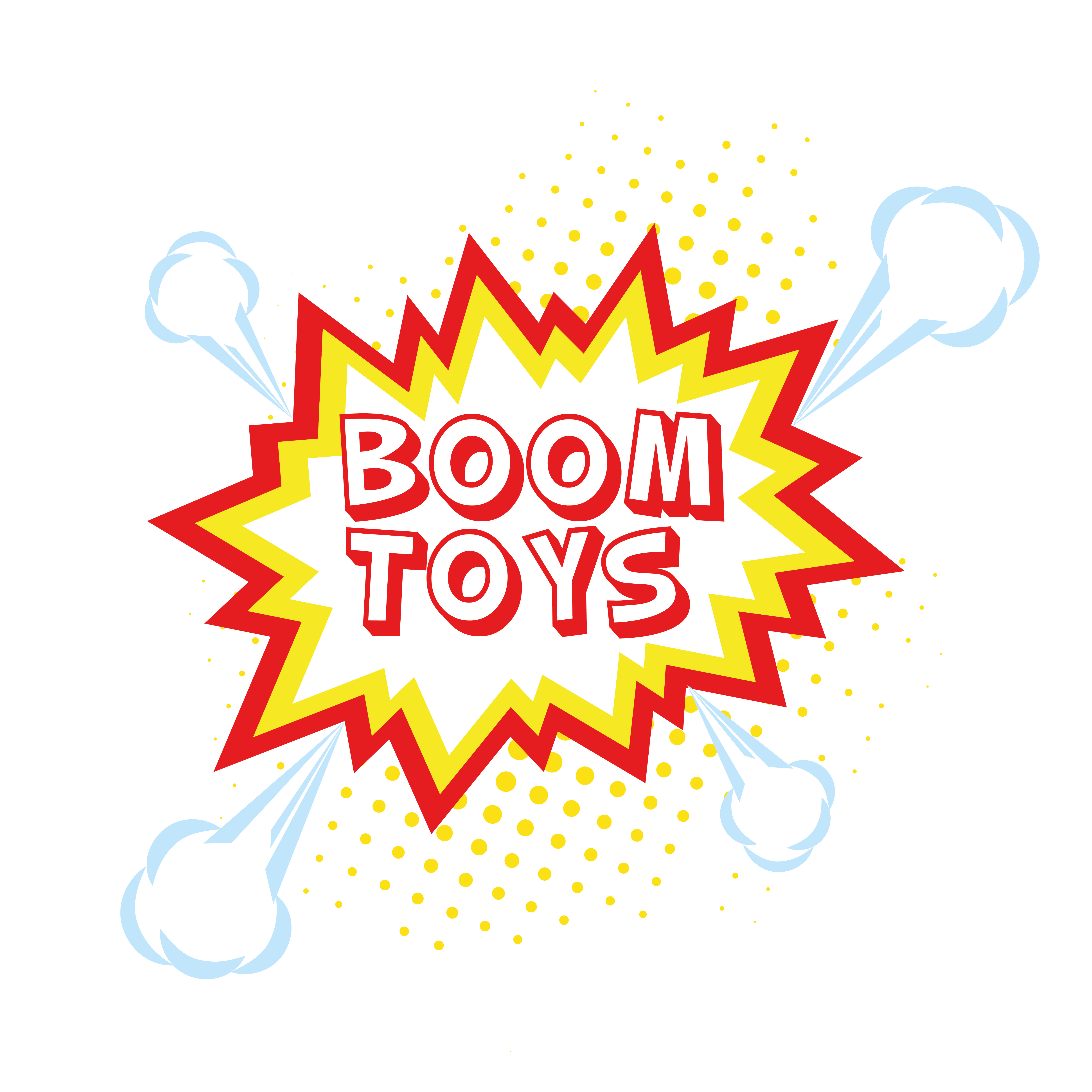
|  |
| --- |
| **Università degli Studi di Salerno Corso di Ingegneria del Software** |

**BoomToys  
ODD - Object Design Document  
Versione 1.0**



Data: 13/12/16

**Partecipanti:**

|  |  |
| --- | --- |
| Nome | Matricola |
| D’Amato Valentina | 0512103052 |
| Russo Daniele | 0512103196 |
| Cicchelli Marco | 0512103292 |
| Sergio Massimo | 0512103070 |

|  |  |
| --- | --- |
| **Scritto da:** | D’Amato, Russo, Cicchelli, Sergio |

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| Data | Versione | Descrizione | Autore |
| 13/12/16 | 0.1 | Creazione documento e stesura punti:  1, 1.1, 1.2, 1.3, 1.4, 2. | Russo D. |
| 15/12/16 | 0.3 | Stesura punto 3 e relativi sottopunti. | Sergio M. |
| 15/12/16 | 0.5 | Stesura indice. | D’Amato V. |
| 16/12/16 | 1.0 | Revisione documento. | Cicchelli, D’Amato, Russo, Sergio. |

Indice

1. **Introduzione4**

1.1 Scelte progettuali dell’Object Design4

1.2 Linee guida della documentazione delle interfacce 5

1.3 Definizioni, Acronimi ed Abbreviazioni 5

1.4 Materiale di riferimento 5

**2. Pacchetti6**

**3. Interfacce delle classi6-22**

**1. INTRODUZIONE**

**1.1 SCELTE PROGETTUALI DELL’OBJECT DESIGN**

Dopo la realizzazione del documento RAD (Requirement Analysis Document) e SDD (System Design Document), abbiamo descritto in linea di massima, quello che sarà il nostro sistema e quindi i nostri obiettivi, tralasciando gli aspetti implementativi.

**Prestazioni VS Costi**

Prendendo in considerazione il sito web che stiamo realizzando, possiamo dire che il non eccessivo budget a nostra disposizione ci ha consentito di realizzare il prodotto utilizzando materiale open source partendo da zero minimizzando così i costi e rendendo l’utilizzo più che soddisfacente. In particolar modo ci siamo serviti di un server web locale gratuito sviluppato dalla Apache Software Foundation.

**Interfaccia vs. Tempo di risposta**

Il tempo di risposta tra server e interfaccia è sufficientemente rapido a soddisfare le esigenze dei vari utenti collegati al sistema. Pertanto la grandezza del database è direttamente proporzionale al tempo di risposta e ricerca nel database.

**Interfaccia vs. Easy-use**

L’interfaccia permette un uso facile dell’intero portale merito delle operazioni intuitive e semplici anche per un’utenza meno esperta, all’utilizzo delle form e alla suddivisione in sezioni dei prodotti. L’interfaccia permette anche un uso facile (Easy-Use) dell’intero sistema che va dalle semplici visualizzazioni alla gestione del sistema di database.

**Costi vs. Mantenimento**

L’utilizzo di risorse open source rendono economici i costi di mantenimento del sistema. Durante lo sviluppo e la progettazione del progetto BoomToys, si è ritenuto opportuno effettuare delle scelte progettuali mirate soprattutto a ridurre la complessità nello sviluppo e a favorire l’usabilità del sito da parte degli utenti finali indipendentemente dal fatto che si trattassero di amministratori o clienti.

La parte server è stata realizzata utilizzando varie tecnologie di programmazione web:

* JSP (JavaServer Pages) in grado di generare pagine web dinamiche
* Servlet capace di gestire le richieste generate da uno o più client

Il database è stato creato in SQL con le relative query sul DBMS grazie all’ausilio del software libero MySQL.

Per la parte client, invece, sono stati utilizzati diversi linguaggi di programmazione:

* HTML ( Hypertext Markup Language) per la struttura statica dei documenti;
* CSS (Cascading Style Sheets) utilizzato per definire lo style delle pagine web;
* Javascript utilizzato per rendere dinamiche le pagine HTML.

**1.2 LINEE GUIDA DELLA DOCUMENTAZIONE DELLE INTERFACCE**

* Le classi hanno nomi singolari;
* I nomi delle classi identificano le operazioni che implementano;
* I metodi sono chiamati con frasi verbali, mentre i campi ed i parametri con frasi sostantivo;

Il sistema è multi-utente, può accedervi chiunque, sia un semplice utente(cliente) e/o un amministratore .  
 Al semplice utente(cliente), il sistema nasconde la logica delle operazioni, fornendogli solamente la consultazione del catalogo dei prodotti presenti e della propria area utente avendo la possibilità di acquistare o prendere visione degli acquisti in quest’ultimi.  
Il sistema permette all’amministratore di gestire tutta la parte amministrativa del sito BoomToys, visionare gli utenti registrati all’interno del portale e inserire e/o cancellare la merce.  
Tutti i tipi di utenti, quindi, saranno coinvolti in query al database ma l’utilizzo delle form rende quest’ultime semplicissime anche per i meno esperti di computer.

**1.3 DEFINIZIONI, ACRONIMI ED ABBREVIAZIONI**

**Acronimi**

* **RAD**: Requirements Analysis Document
* **SDD**: System Design Document
* **ODD**: Object Design Document
* **DBMS**: DataBase Management System

**Definizioni**

* **Amministratore:** responsabile della gestione dell’intero sito BoomToys
* **Utente/Cliente:** qualsiasi persona registrata;
* **Query:** In [informatica](http://it.wikipedia.org/wiki/Informatica) il termine ***query*** viene utilizzato per indicare l'interrogazione da parte di un utente di un [database](http://it.wikipedia.org/wiki/Database), strutturato tipicamente secondo il [modello relazionale](http://it.wikipedia.org/wiki/Modello_relazionale), per compiere determinate operazioni sui [dati](http://it.wikipedia.org/wiki/Dati) (selezione, inserimento, cancellazione dati, aggiornamento ecc.).
* **SQL** è un linguaggio standardizzato per database basati sul modello relazionale in grado di poter:
* Creare e modificare schemi di database
* Inserire, cancellare e modificare dati memorizzati
* Interrogare dati memorizzati
* Gestire gli utenti e i permessi
* **MySQL** è un software per la gestione di database relazionali composto da un client a riga di comando e un server.
  1. **RIFERIMENTI**
* Libro di testo “Object-Oriented Software Engineering –Using UML, Patterns and Java” di Bernd Bruegge e Allen H. Dutoit;
* Documentazione online;
* Materiale del docente reperibile sulla piattaforma el-platform;
* Consultazione di sistemi web, di proprietà di terze parti (quale Toys Center e Lego), già esistenti e che si riferiscono allo stesso ambito di mercato del presente progetto;
* Manuali e libri di testo dei principali linguaggi di programmazione (HTML, CSS, Java, SQL) dedicati allo sviluppo di sistemi per il Web.
* RAD
* SDD

1. **PACCHETTI**

Per quanto riguarda l’organizzazione dei file per il progetto BoomToys è stato scelto di effettuare una prima suddivisione tra i documenti che riguardano la struttura dell’amministratore da quella degli utenti. In secondo luogo i documenti sorgenti sono stati divisi in base al linguaggio di programmazione utilizzato.  
Quindi abbiamo cartelle che contengono le immagini utilizzate all’interno del sito web e ulteriori cartelli per gli script javascript, fogli di stile e cartoline.

1. **INTERFACCIA DELLE CLASSI**
   1. **SOTTOSISTEMA GESTIONE UTENTI**

**public** **class** UserBean {

**private** **int** numUtente;

**private** String id;

**private** String pass;

**public** UserBean() {

**this**.numUtente = 0; /\* 16 caratteri \*/

**this**.id = "";

**this**.pass = "";

}

**public** **int** getNumUtente() {

**return** numUtente;

}

**public** **void** setNumUtente(**int** numUtente) {

**this**.numUtente = numUtente;

}

**public** String getId() {

**return** id;

}

**public** **void** setId(String id) {

**this**.id = id;

}

**public** String getPass() {

**return** pass;

}

**public** **void** setPass(String pass) {

**this**.pass = pass;

}

@Override

**public** String toString() {

**return** "UserBean [numUtente=" + numUtente + ", id=" + id + ", pass=" + pass + "]";

}

}

**3.2 SOTTOSISTEMA GESTIONE PRODOTTI**

**public** **class** MerceBean {

**private** **int** codMerce;

**private** String nome;

**private** String tipo;

**private** **double** costo;

**private** **int** codice;

**private** **int** codiceOrdine;

**public** MerceBean() {

**this**.codMerce = 0;

**this**.nome = "";

**this**.tipo = "";

**this**.costo = 0;

**this**.codice = 0;

**this**.codiceOrdine = 0;

}

**public** **int** getCodMerce() {

**return** codMerce;

}

**public** **void** setCodMerce(**int** codMerce) {

**this**.codMerce = codMerce;

}

**public** String getNome() {

**return** nome;

}

**public** **void** setNome(String nome) {

**this**.nome = nome;

}

**public** String getTipo() {

**return** tipo;

}

**public** **void** setTipo(String tipo) {

**this**.tipo = tipo;

}

**public** **double** getCosto() {

**return** costo;

}

**public** **void** setCosto(**double** costo) {

**this**.costo = costo;

}

**public** **int** getCodice() {

**return** codice;

}

**public** **void** setCodice(**int** codice) {

**this**.codice = codice;

}

**public** **int** getCodiceordine() {

**return** codiceOrdine;

}

**public** **void** setCodiceOrdine(**int** codiceOrdine) {

**this**.codiceOrdine = codiceOrdine;

}

@Override

**public** String toString() {

**return** "MerceBean [cod\_merce=" + codMerce + ", nome=" + nome + ", tipo=" + tipo + ", costo=" + costo

+ ", codice=" + codice + ", codiceOrdine=" + codiceOrdine + "]";

}

}

**3.3 SOTTOSISTEMA GESTIONE CARRELLO**

**public** **class** CartBean {

**private** List<MerceBean> products;

**public** CartBean(){

products = **new** ArrayList<MerceBean>();

}

**public** **void** addMerceBean(MerceBean product) {

products.add(product);

}

**public** **void** deleteVinylBean(MerceBean product) {

**for**(MerceBean prod : products) {

**if**(prod.getCodMerce() == product.getCodMerce()) {

products.remove(prod);

**break**;

}

}

}

**public** **boolean** thereAre(MerceBean product){

**for**(MerceBean prod : products) {

**if**(prod.getCodMerce() == product.getCodMerce()) {

**return** **true**;

}

}

**return** **false**;

}

**public** List<MerceBean> getCart() {

**return** products;

}

}

**3.4 SOTTOSISTEMA GESTIONE ORDINI**

**package** bean;

**public** **class** **OrderBean** {

**private** **int** codOrdine;

**private** **String** dataOrdine;

**private** **int** nMerci;

**private** **double** total;

**private** **String** metodoDiPagamento;

**private** **String** numCard;

**private** **int** numUtente;

**public** **OrderBean**() {

**this**.codOrdine = 0;

**this**.dataOrdine = "";

**this**.nMerci = 0;

**this**.total = 0;

**this**.metodoDiPagamento = "";

**this**.numCard = "";

**this**.numUtente = 0;

}

**public** **int** **getCodOrdine**() {

**return** codOrdine;

}

**public** **void** **setCodOrdine**(**int** codOrdine) {

**this**.codOrdine = codOrdine;

}

**public** **String** **getDataOrdine**() {

**return** dataOrdine;

}

**public** **void** **setDataOrdine**(**String** dataordine) {

**this**.dataOrdine = dataordine;

}

**public** **int** **getnMerci**() {

**return** nMerci;

}

**public** **void** **setnMerci**(**int** nMerci) {

**this**.nMerci = nMerci;

}

**public** **double** **getTotal**() {

**return** total;

}

**public** **void** **setTotal**(**double** total) {

**this**.total = total;

}

**public** **String** **getMetodoDiPagamento**() {

**return** metodoDiPagamento;

}

**public** **void** **setMetodoDiPagamento**(**String** metodoDiPagamento) {

**this**.metodoDiPagamento = metodoDiPagamento;

}

**public** **String** **getNumCard**() {

**return** numCard;

}

**public** **void** **setNumCard**(**String** numCard) {

**this**.numCard = numCard;

}

**public** **int** **getNumUtente**() {

**return** numUtente;

}

**public** **void** **setNumUtente**(**int** numUtente) {

**this**.numUtente = numUtente;

}

***@Override***

**public** **String** **toString**() {

**return** "OrderBean [codOrdine=" + codOrdine + ", dataordine=" + dataOrdine + ", nMerci=" + nMerci + ", total="

+ total + ", metodoDiPagamento=" + metodoDiPagamento + ", numCard=" + numCard + ", numUtente="

+ numUtente + "]";

}

}