Università degli Studi di Salerno

Corso di Ingegneria del Software

StarBay
ODD
Versione 3.0

16/12/2016

Coordinatore del progetto:

Nome	Matricola

Partecipanti:

Nome	Matricola
Roberto Contaldo	0512102948
Gianluca De Luca Fiscone	0512103290

Scritto da:	Roberto Contaldo,Gianluca De Luca Fiscone

Revision History

Data	Versione	Descrizione	Autore
10/12/2016	1.0	Prima Stesura.	Roberto Contaldo, Gianluca De Luca Fiscone
16/12/2016	2.0	Documentazione del riuso: Componenti Off the Shelf. Packages. Class Interfaces.	Roberto Contaldo, Gianluca De Luca Fiscone
16/12/2016	2.0	Revisione e consegna.	Roberto Contaldo
16/12/2016	3.0	Linee guida per la documentazione delle interfacce: modifica. Packages: modifica.	Roberto Contaldo, Gianluca De Luca Fiscone

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 StarBay, tuttavia esso andrà ad impattare sulle prestazioni del sistema perché l'avvio del server, da parte di OpenShift, 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:

```java
/**
 * Cliente
 * Classe che descrive un cliente
 */

package bean;

public class Cliente
{
    private String nome;
    private String cognome;
    private String comune;
    private String username;
    private String password;
    private String indirizzo;
    private String iban;

    public String getNome()
    {
        return nome;
    }

    public void setNome(String nome)
    {
        this.nome = nome;
    }

    // ...
}
```

```java
/**
 * LoginGestore
 * Classe che controlla i dati e reindirizza il gestore alla sua home page
 */

package servlet;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;


@WebServlet("/login-gestore")
public class LoginGestore extends HttpServlet
{
    public void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
    {
        // ...
    }

    public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
    {
        // ...
    }
}
```

1.2.2 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.

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

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

1.2.3 Pagine HTML

Le pagine HTML, statiche e dinamiche, devono essere totalmente aderenti allo standard HTML versione 5. Le pagine HTML, devono utilizzare l'indentazione, per facilitare la lettura, secondo le seguenti regole:

- Un'indentazione consiste in una tabulazione;
- Ogni tag deve avere un'indentazione maggiore del tag che lo contiene
- Sebbene l'uso dell'indentazione sia arbitrario, ogni tag di chiusura deve avere lo stesso livello di indentazione del corrispondente tag di apertura;
- I tag di commento devono seguire le stesse regole che si applicano ai tag normali;

1.2.4 Script Javascript

Gli script che svolgono funzioni distinte dal mero rendering della pagina sono collocati all'interno delle pagine HTML. 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) {
```

1.2.5 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.

1.2.6 Database SQLite

Le tabelle del database dovrebbero essere in terza forma normale di Codd (3NF). I nomi delle tabelle devono seguire le seguenti regole:

- sono costituiti da sole lettere maiuscole;

- se il nome è costituito da più parole, queste sono separate da un underscore (_);

- il nome deve essere un sostantivo plurale tratto dal dominio del problema ed esplicativo del contenuto.

I nomi dei campi devono seguire le seguenti regole:

- sono costituiti da sole lettere;

- se il nome è costituito da più parole, la prima è in minuscolo, le successive in maiuscolo;

- il nome deve essere un sostantivo singolare tratto dal dominio del problema ed esplicativo del contenuto.

1.3 Definizioni, acronimi e abbreviazioni

Acronimo	Descrizione
RAD	Requirement Analysis Document
SDD	System Design Document
ODD	Object Design Document
DMBS	Data Base Management System
SQL	Structured Query Language
HW	Hardware
SW	Software
GUI	Graphical User Interface

1.4 References

- Bernd Bruegge e Allen H. Dutoit - Object-Oriented Software Engineering (using UML, Patterns and JavaTM) – Prentice Hall
- Sommerville, Software Engineering – Addison Wesley
- Object-Oriented Software Engineering (using UML, Patterns and JavaTM)
- StarBay_RAD_4.0.pdf
- StarBay_SDD_4.0.pdf

2 Documentazione del riuso

2.1 Componenti Off the Shelf

Le componenti Off the Shelf che saranno utilizzate per l'implementazione del sito sono:

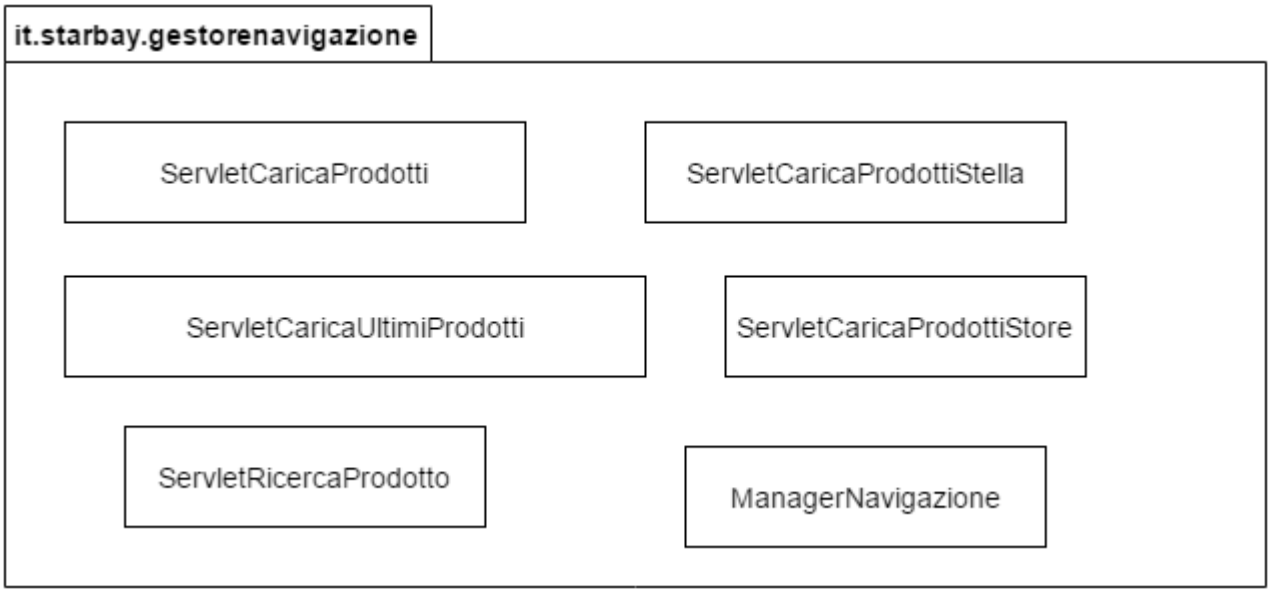
- DBMS
- Bootstrap

Il DBMS è un sistema software progettato per consentire la creazione, la manipolazione (da parte di un amministratore DBA) e l'interrogazione efficiente (da parte di uno o più utenti client) di database.
Bootstrap, invece, è una raccolta di strumenti liberi per la creazione di siti e applicazioni per il Web. Essa contiene modelli di progettazione basati su HTML e CSS, sia per la tipografia, che per le varie componenti dell'interfaccia, come moduli, pulsanti e navigazione, così come alcune estensioni opzionali di JavaScript.

3 Packages

Di seguito verranno mostrati i package relative ad ogni gestione, specificando le funzionalità che loro supportano.

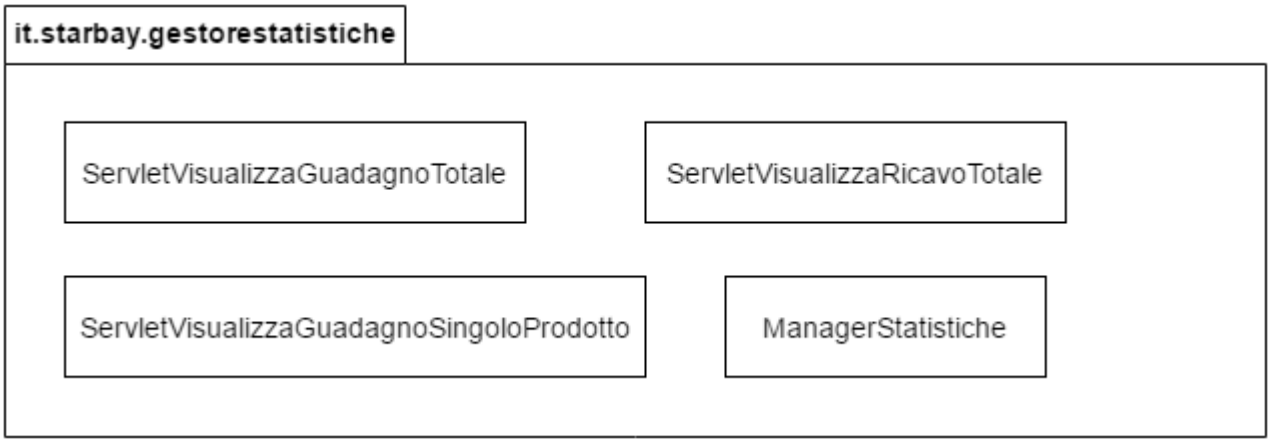
3.1 PCK1 – it.starbay.gestionenavigazione (Gestione Navigazione)



Il package visualizzato sopra è relativo alla gestione della navigazione.
Le servlet "ServletCaricaProdottiStella", "ServletCaricaUltimiProdotti",

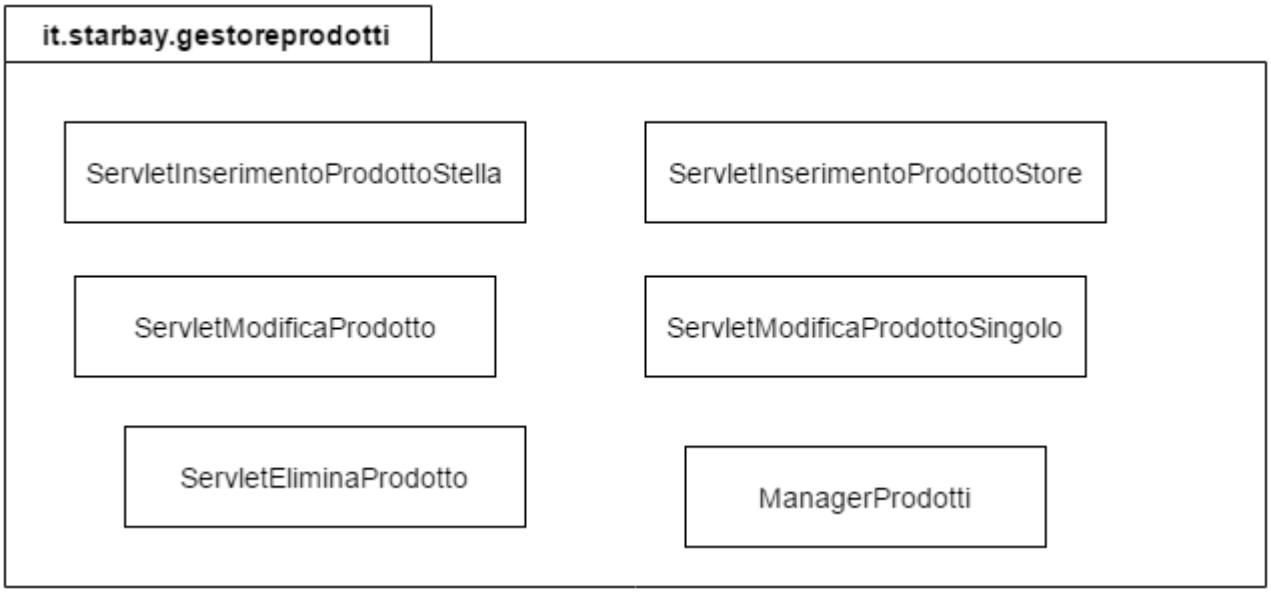
“ServletCaricaProdottiStore”, verranno richiamate dalle pagine .jsp corrispondenti a stelle.jsp, index.jsp e store,jsp in cui verranno visualizzati i vari cataloghi presenti nel database. La servlet “ServletCaricaProdotti” verrà richiamata, invece dalle jsp modifica_prodotto.jsp e eliminazione_prodotto.jsp. In ogni servlet avremo la creazione di un ManagerNavigazione che avrà dei metodi, descritti nel javadoc, in cui verranno eseguite le query e poi il loro risultato verrà processato dalla servlet stessa. Infine, la servlet “ServletRicercaProdotto” verrà richiamata quando l’utente proverà ad effettuare una ricerca tramite una barra posta in ogni pagina principale di StarBay. La servlet richiamerà una delle funzioni di ManagerNavigazione che permetterà di eseguire la query, passandogli come parametro la chiave da ricercare.

3.2 PCK2 – it.starbay.gestionestatistiche (Gestione Statistiche)



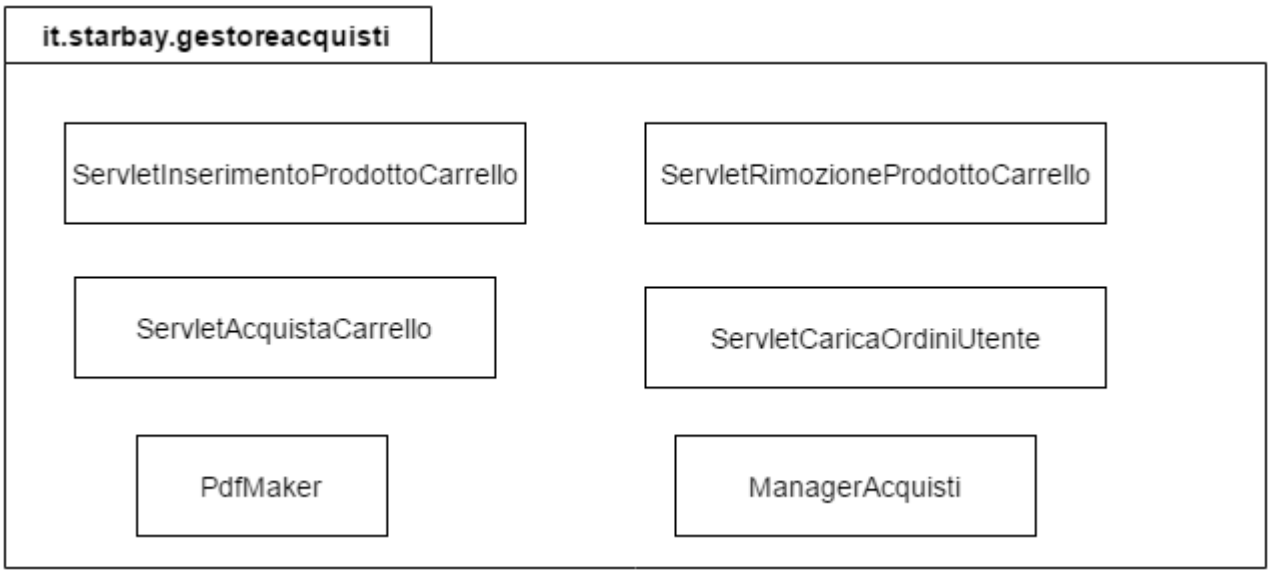
Il package visualizzato sopra è relativo alla gestione delle statistiche.
Le servlet “ServletVisualizzaGuadagnoTotale”, “ServletVisualizzaRicavoTotale”, “ServletVisualizzaGuadagnoSingoloProdotto”, verranno richiamate dalle pagine .jsp corrispondenti a visualizza_guadagno_totale.jsp, visualizza_ricavo_totale.jsp e visualizza_guadagno_singolo_prodotto.jsp in cui verranno visualizzati i valori in euro. In ogni servlet avremo la creazione di un ManagerStatistiche che avrà dei metodi, descritti nel javadoc, in cui verranno eseguite le query, in seguito il loro risultato verrà processato dalla servlet stessa e visualizzato all’utente.

3.3 PCK3 – it.starbay.gestioneprodotti (Gestione Prodotti)



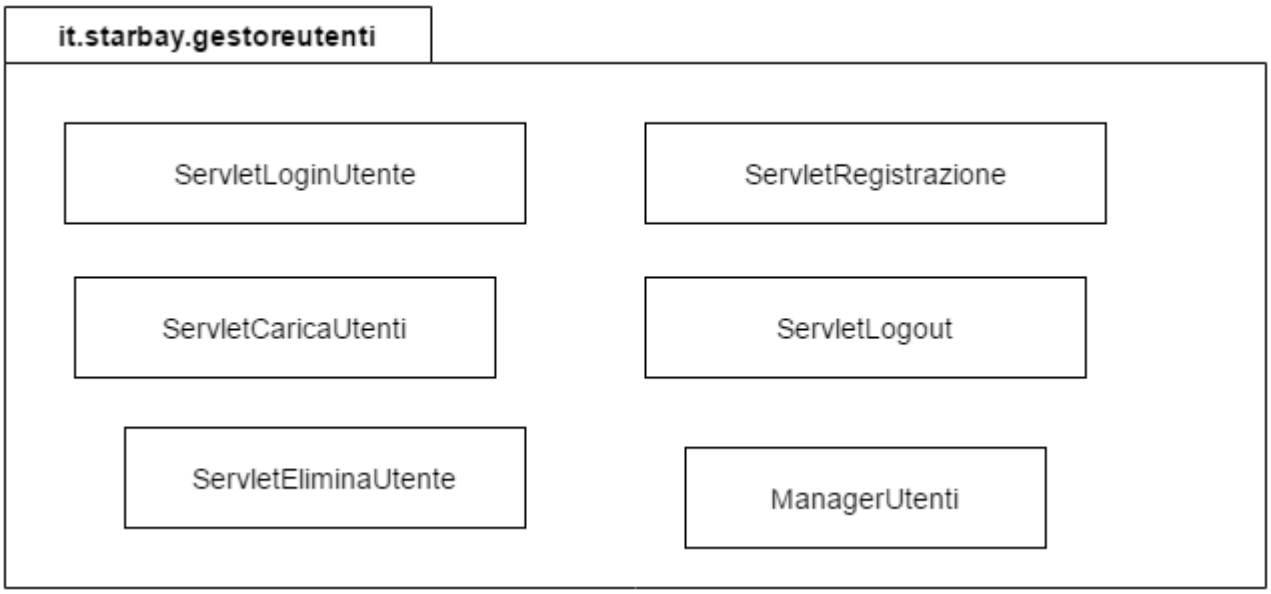
Il package visualizzato sopra è relativo alla gestione dei prodotti.
Le servlet "ServletInserimentoProdottoStella", "ServletInserimentoProdottoStore", "ServletModificaProdotto", "ServletModificaProdottoSingolo", "ServletEliminaProdotto" verranno richiamate dalle pagine .jsp corrispondenti a inserimento_prodotto_stella.jsp, inserimento_prodotto_store.jsp, modifica_prodotto.jsp e modifica_prodotto_singolo,jsp, in cui verranno visualizzate delle tabelle contenenti i prodotti o un form con degli input da completare. In ogni servlet avremo la creazione di un ManagerProdotti che avrà dei metodi, descritti nel javadoc, in cui verranno eseguite le query, che permetteranno l'inserimento, la modifica e l'eliminazione dei prodotti.

3.4 PCK4 – it.starbay.gestioneacquisti (Gestione Acquisti)



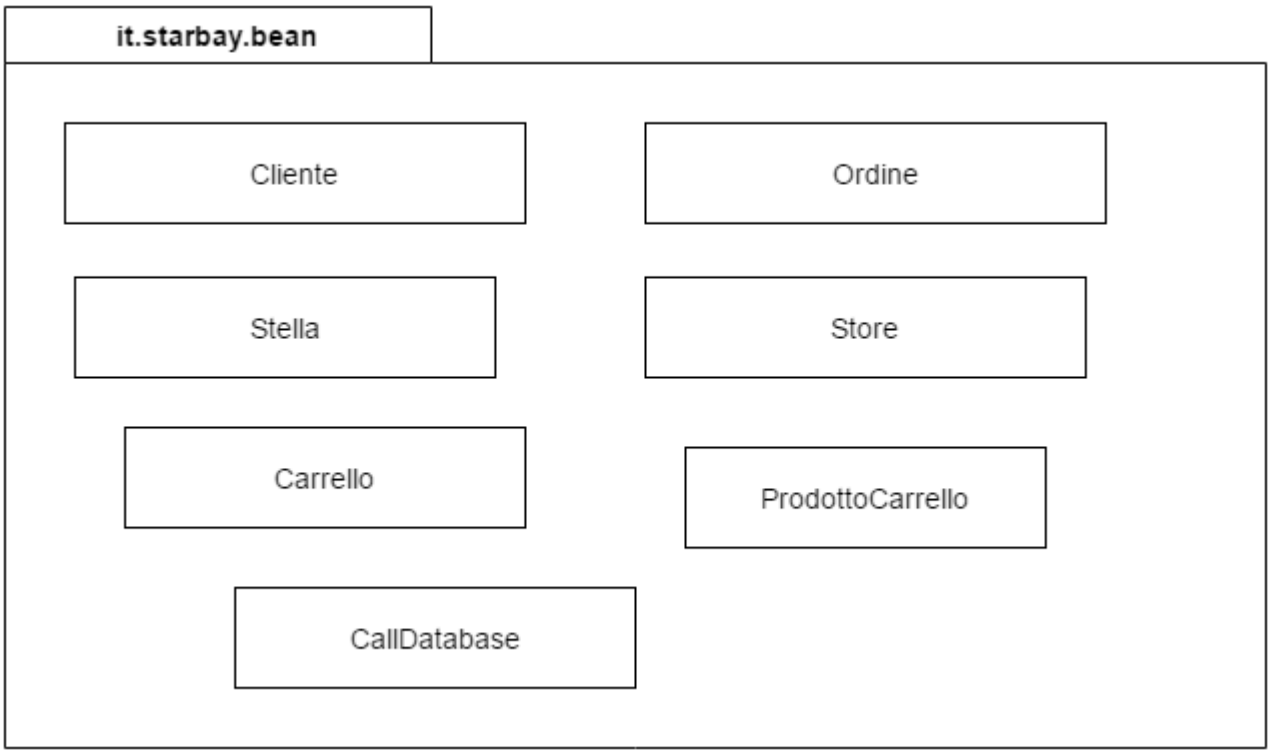
Il package visualizzato sopra è relativo alla gestione degli acquisti.
Le servlet “ServletInserimentoProdottoCarrello”, verrà richiamata ogni qualvolta che si cercherà, nelle pagine relative alla visualizzazione del catalogo, di aggiungere un prodotto al carrello. La servlet “ServletRimozioneProdottoCarrello” verrà richiamata, nella pagina carrello.jsp, nel momento in cui l’utente cliccherà sull’icona di eliminazione. La servlet “ServletCaricaOrdiniUtente” verrà chiamata quando il cliente entrerà nel suo profilo, immediatamente dopo verrà chiamata la classe PdfMaker che creerà, per ogni ordine di una stella, un file pdf in cui ci saranno i dettagli dell’ordine effettuato. Infine, la servlet “ServletAcquistaCarrello” verrà richiamata nel momento in cui l’utente effettuerà l’acquisto dei prodotti presenti nel carrello. In tal momento, la servlet creerà un ManagerAcquisti che si occupa di eseguire le query per memorizzare l’ordine nel database.

3.5 PCK5 – it.starbay.gestioneutenti (Gestione Utenti)



Il package visualizzato sopra è relativo alla gestione degli utenti.
Le servlet “ServletLoginUtente” e “ServletLogout” verranno richiamate nel momento in cui l’utente si logga e si disconnette al sistema, mentre la servlet “ServletRegistrazione” verrà richiamata quando un visitatore si appresta, attraverso un form, a registrarsi. Infine, le servlet “ServletEliminaUtente” e “ServletCaricaUtenti” verranno richiamate quando l’amministratore tenterà di eliminare un utente.
Ogni servlet, tranne la servlet per il logout, creerà un ManagerUtenti che eseguirà le query al database permettendo così al control di svolgere le proprie funzionalità.

3.6 PCK6 – it.starbay.gestionebean (Bean)



Il package visualizzato sopra è relativo ai Bean presenti nella piattaforma.
Sono stati estratti dai class diagram insieme ai loro attributi. Ogni Bean quindi rappresenterà una classe che avrà un costruttore e i metodi get e set.

4 Class Interfaces

Il class interfaces sarà realizzato tramite l'applicativo javadoc, utilizzato per la generazione automatica della documentazione del codice sorgente scritto in linguaggio Java. Nella cartella "deliverables" sarà presente il file di javadoc per l'interfaccia delle classi e la relativa suddivisione in pacchetti.

5 Glossario

TERMINE	DESCRIZIONE
Classe	Astrazione che specifica lo stato ed il comportamento di un insieme di oggetti.
Costruttore	Una operazione che crea un oggetto e inizializza il suo stato.
JavaDoc	Strumento che estrae dai commenti di un programma una documentazione dettagliata del codice.
Metodi	Funzioni e procedure che operano sui dati di un oggetto. I programmi dovrebbero interagire con i dati di una classe solo attraverso i suoi metodi.
Package	Costrutto Java che fornisce un raggruppamento di un insieme di classi e interfacce in relazione tra loro.
Parametro	Una variabile passata ad un metodo quando viene chiamato. In Java si utilizzano solo chiamate per valore, cioè, il metodo prende una copia di tutti i valori dei parametri e non può modificarli.
Signature	È la "firma" di un metodo, è infatti costituita da nome dell'operazione, lista completa dei parametri e tipo del valore di ritorno.