Eleworld2018

Object Design Document

Daniele Cioffi

Mario Consalvo

Indice

Coordinatori del progetto

Prof. Andrea DE LUCIA
Prof.ssa Rita FRANCESCE

Partecipanti

Daniele Cioffi – Matricola 0512103446
Mario Consalvo – Matricola 0512103500

Cronologia delle Revisioni

Data	Versione	Descrizione	Autore
08/01/2018	1.0	Prima Stesura	Cioffi Daniele Mario Consalvo
12/01/2018	1.1	Aggiunta tabelle	Cioffi Daniele Mario Consalvo
20/01/2018	1.2	Aggiunta diagrammi	Cioffi Daniele Mario Consalvo

1. Introduzione

1.1 Object Design Trade-off

Dopo la realizzazione dei documenti RAD e SDD abbiamo descritto il nostro sistema lasciando fuori soltanto gli aspetti implementativi. Il seguente documento ha lo scopo di produrre un modello capace di integrare in modo coerente e preciso tutte le funzionalità individuate nelle fasi precedenti. In particolare definisce le interfacce delle classi, le operazioni, i tipi, gli argomenti e la signature dei sottosistemi definiti nel System Design. Inoltre sono specificati i seguenti trade-off.

Comprensibilità vs Tempo:

Il codice deve essere quanto più comprensibile possibile per facilitare la fase di testing ed eventuali future modifiche. Il codice sarà quindi accompagnato da commenti che ne semplifichino la comprensione. Ovviamente questa caratteristica aggiungerà un incremento di tempo allo sviluppo del nostro progetto.

Prestazioni vs Costi:

Essendo il progetto sprovvisto di budget, al fine di mantenere prestazioni elevate, per alcune funzionalità verranno utilizzati dei template open source esterni in particolare Bootstrap.

Interfaccia vs Usabilità:

L'interfaccia grafica è stata realizzata in modo da essere molto semplice, chiara e concisa, fa uso di form e pulsanti disposti in maniera da rendere semplice l'utilizzo del sistema da parte dell'utente finale.

Sicurezza vs Efficienza:

La sicurezza, come descritto nei requisiti non funzionali del RAD, rappresenta uno degli aspetti importanti del sistema. Tuttavia, dati i tempi di sviluppo molto limitati, ci limiteremo ad implementare sistemi di sicurezza basati su username e password degli utenti.

1.2 Definizioni, acronimi e abbreviazioni

Acronimi:

- RAD: Requirements Analysis Document
- SDD: System Design Document
- ODD: Object Design Document

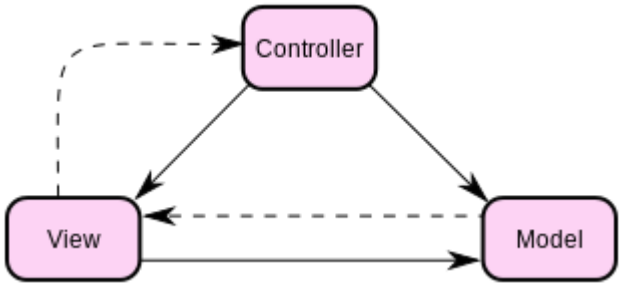
Abbreviazioni:

- DB: DataBase

2. Package

Utilizzando il modello MVC (Model View Control), il progetto è composto da 3 package, uno per ogni componente del modello. Abbiamo quindi un package di pagine jsp, uno di servlet(java), e un altro di pagine Java.

Il Model-View-Controller, è un pattern architetturale molto diffuso nello sviluppo di sistemi software, in particolare nell'ambito della programmazione orientata agli oggetti, in grado di separare la logica di presentazione dei dati dalla logica di business. Questo pattern si posiziona nel livello di presentazione in una Architettura multi-tier. Il componente centrale del MVC, il modello, cattura il comportamento dell'applicazione in termini di dominio del problema, indipendentemente dall'interfaccia utente. Il modello gestisce direttamente i dati, la logica e le regole dell'applicazione. Una vista può essere una qualsiasi rappresentazione in output di informazioni, come un grafico o un diagramma. Sono possibili viste multiple delle stesse informazioni, come ad esempio un grafico a barre per la gestione e la vista tabellare per l'amministrazione. La terza parte, il controller, accetta l'input e lo converte in comandi per il modello e/o la vista.



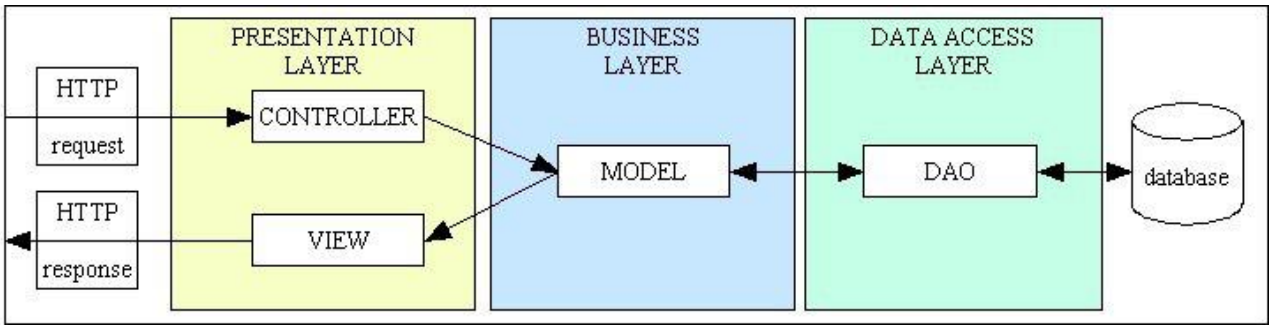
2.1 Package Model

La parte del model è responsabile della memorizzazione e dell'aggiornamento dei dati. In questo sistema, il model è rappresentato da 3 tipologie di classi:

- Bean
- DAO
- DAOInterface

Le classi bean, rappresentano l'oggetto persistente vero e proprio, dialogano attraverso le classi DAO con il database, utilizzando un'interfaccia apposita. Nell'ambito della programmazione Web, il DAO (Data Access Object) è un pattern architetturale per la gestione della persistenza: si tratta fondamentalmente di una classe con relativi metodi che rappresenta un'entità tabellare di un RDBMS, usata principalmente in applicazioni web per stratificare e isolare l'accesso ad una tabella tramite query creando un maggiore livello di astrazione ed una più facile manutenibilità. I metodi del DAO con le rispettive query verranno così richiamati dalle classi della business logic. Inoltre sono state create delle interfacce in modo tale da rendere i metodi per l'accesso al database indipendenti dall'implementazione.

Di seguito è riportata una tabella di descrizione delle classi appartenenti al package "Model":



CLASSE	DESCRIZIONE
CARRELLOBEAN.JAVA	Descrive un carrello
CARRELLODAO.JAVA	Descrive un carrello dal punto di vista del DB
CARRELLODAOINTERFACE.JAVA	Descrive l'interfaccia della classe carrello
CARTADICREDITOBEAN.JAVA	Descrive una carta di credito
CARTADICREDITODAO.JAVA	Descrive una carta di credito dal punto di vista del DB
CARTADICREDITODAOINTERFACE.JAVA	Descrive l'interfaccia della classe carta di credito
CATEGORIABEAN.JAVA	Descrive una categoria di prodotto
CATEGORIADAO.JAVA	Descrive una categoria di prodotto dal punto di vista del DB
CATEGORIADAOINTERFACE.JAVA	Descrive l'interfaccia della classe categoria
DETTAGLIODRINIDAOINTERFACE.JAVA	Descrive l'interfaccia della classe dettagli ordine
DETTAGLIORDINIBEAN.JAVA	Descrive i dettagli ordine
DETTAGLIORDINIDAO.JAVA	Descrive i dettagli ordine dal punto di vista del DB
DRIVERMANAGERCONNECTIONPOOL.JAVA	Descrive la connessione con il DB
ORDINEBEAN.JAVA	Descrive un ordine di prodotti
ORDINEDAO.JAVA	Descrive un ordine di prodotti dal punto di vista del DB
ORDINEDAOINTERFACE.JAVA	Descrive l'interfaccia della classe ordine
PRODOTTIBEAN.JAVA	Descrive un prodotto
PRODOTTIDAO.JAVA	Descrive un prodotto dal punto di vista del DB
PRODOTTIDAOINTERFACE.JAVA	Descrive l'interfaccia della classe prodotti
UTENTEBEAN.JAVA	Descrive un utente
UTENTEDAO.JAVA	Descrive un utente dal punto di vista del DB
UTENTEDAOINTERFACE.JAVA	Descrive l'interfaccia della classe utente

2.2 Package View

Le classi di visualizzazione rappresentano le rappresentazioni visive del modello. Tutte le viste dovranno memorizzare informazioni comuni e rispondere a determinati eventi attraverso metodi. Nel nostro caso, parliamo di pagine jsp, JavaServer Pages, cioè una tecnologia di programmazione Web in Java, per lo sviluppo della logica di presentazione (tipicamente secondo il pattern MVC) di applicazioni Web. Esse forniscono contenuti dinamici in formato HTML o XML. La tecnlogia JSP si basa su un insieme di speciali tag, all'interno di una pagina HTML, con cui possono essere invocate funzioni predefinite sotto forma di codice Java (e/o funzioni JavaScript). Sono state create quindi, pagine JSP specifiche per ogni situazione.

Di seguito la descrizione delle JSP:

CLASSE	DESCRIZIONE
ACCOUNT.JSP	View che riguarda la pagina di registrazione
ACCOUNTINFO.JSP	View che riguarda la pagina di informazioni agguntive utente
CANCELLA.JSP	View che riguarda la pagina di eliminazione prodotti dal magazzino
CHECKOUT.JSP	View che riguarda la pagina dei prodotti nel carrello
CHISIAMO.JSP	View che riguarda la pagina di informazioni del sito
CONTACT.JSP	View che riguarda la pagina dei contatti del sito
FOOTER.JSP	View che riguarda il footer del front-end
FOOTER2.JSP	View che riguarda il footer del back-end
HEADER.JSP	View che riguarda l'header del front-end
HEADER2.JSP	View che riguarda l'header del back-end
INDEX.JSP	View che riguarda l'index del front-end
INDEX2.JSP	View che riguarda l'index del back-end
INFORMAZIONI.JSP	View che riguarda la pagina di inserimento informazioni aggiuntive utente
INSERISCI.JSP	View che riguarda la pagina di inserimento prodotti nel magazzino
LOGIN.JSP	View che riguarda il login
MODIFICA.JSP	View che riguarda la pagina di modifica prodotti nel magazzino
OFFERTE.JSP	View che riguarda la pagine delle offerte
PAGAMENTO.JSP	View che riguarda la pagina successiva al pagamento
PASSWORDLOST.JSP	View che riguarda la pagina di password persa
PRODUCT.JSP	View che riguarda la pagina di visualizzazione prodotti
SINGLE.JSP	View che riguarda la pagina del singolo prodotto

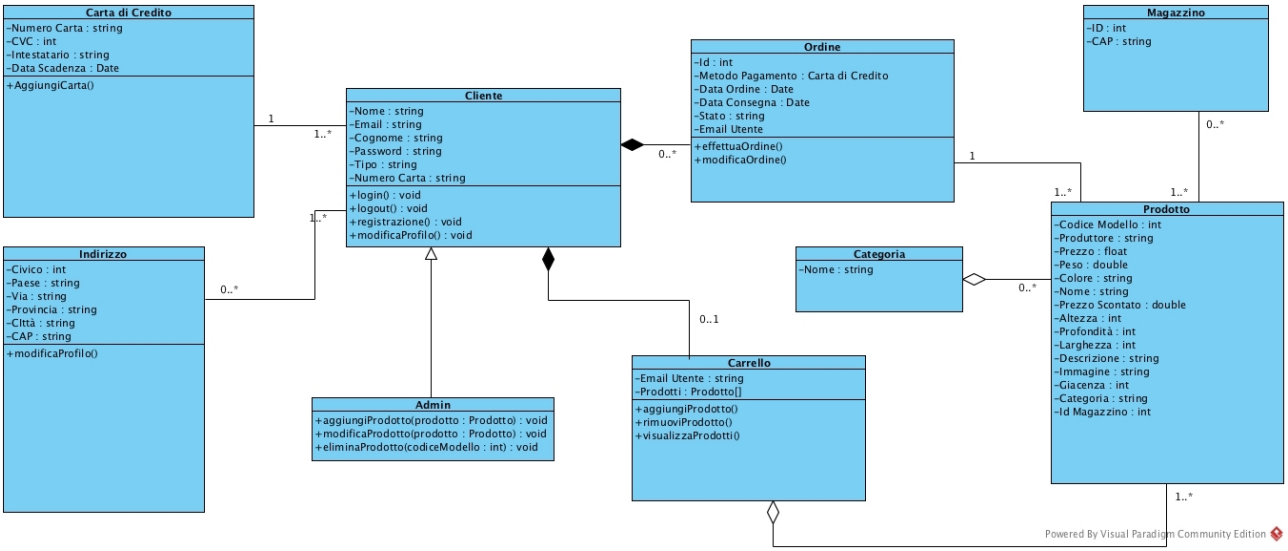
2.3 Package Control

Il controller è uno dei componenti più importanti in un'applicazione MVC poiché funge da coordinatore tra la vista e il modello e gestisce anche le richieste degli utenti. La richiesta dell'utente viene ricevuta dal controller che richiede al modello di eseguire diverse operazioni, e quindi decide il risultato da tornare all'utente. In MVC esiste una relazione disaccoppiata tra l'URL richiesto e la classe controller utilizzata per soddisfare la richiesta. Il client effettua una richiesta utilizzando un URL ed è responsabilità dell'applicazione MVC passare la richiesta al componente appropriato o al controller che gestirà la richiesta. Questo disaccoppiamento semplifica la modifica della struttura dell'URL in quanto l'utente non è a conoscenza del file che effettivamente serve alla richiesta. Ciò è anche positivo dal punto di incapsulamento poiché i dettagli interni dell'applicazione non sono esposti al mondo esterno. Nel nostro caso, sono state create delle classi Java che operano all’interno di un server web, le servlet. L'uso più frequente delle servlet è la generazione di pagine web dinamiche a seconda dei parametri di richiesta inviati dal client browser dell'utente al server.

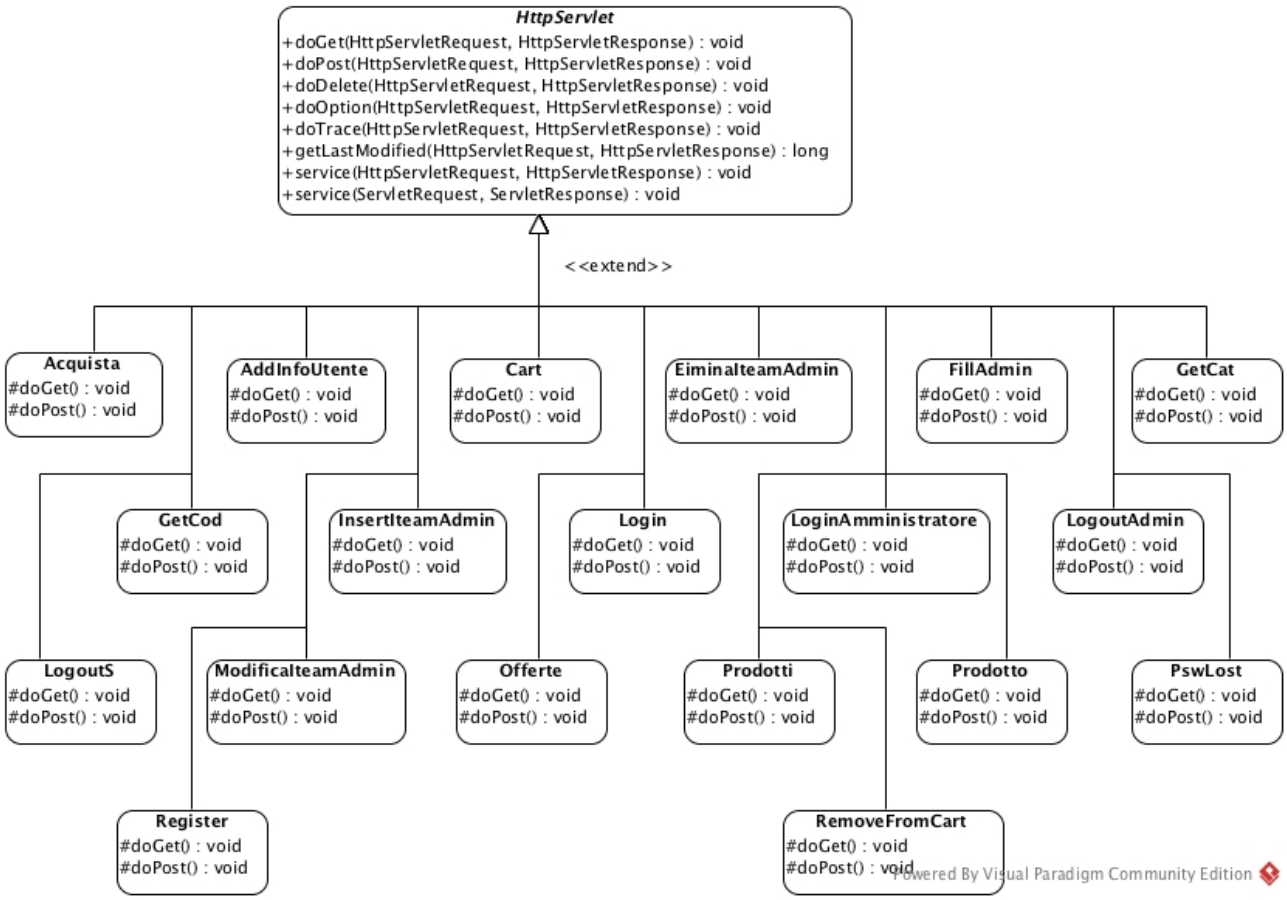
Di seguito sono riportate le descrizioni delle servlet:

CLASSE	DESCRIZIONE
ACQUISTA.JAVA	Controller che permette l'acquisto di un prodotto
ADDINFOUTENTE.JAVA	Controller che permette di aggiungere informazioni all'entità utente
CART.JAVA	Controller che permette la gestione del carrello
EIMINAITEAMADMIN.JAVA	Controller che permette l'eliminazione dei prodotti dal carrello
FILLADMIN.JAVA	Chiamata AJAX per consultare il DB
GETCAT.JAVA	Chiamata AJAX per consultare il DB
GETCOD.JAVA	Chiamata AJAX per consultare il DB
INSERTITEAMADMIN.JAVA	Controller che permette l'inserimento di un nuovo prodotto
LOGIN.JAVA	Controller che permette la gestione del login
LOGINAMMINISTRATORE.JAVA	Controller che permette la gestione del login dell'amministratore
LOGOUTADMIN.JAVA	Controller che permette la gestione del logout dell'amministratore
LOGOUTS.JAVA	Controller che permette la gestione del logout
MODIFICAITEAMADMIN.JAVA	Controller che permette la modifica di un prodotto
OFFERTE.JAVA	Controller che permette la creazione dinamica della pagina offerte.jsp
PRODOTTI.JAVA	Controller che permette la creazione dinamica della pagina prodotti.jsp
PRODOTTO.JAVA	Controller che permette la creazione dinamica della pagina single.jsp
PSWLOST.JAVA	Controller che permette il cambio della password in caso di smarriento
REGISTER.JAVA	Controller che permette la creazione del profilo utente
REMOVEFROMCART.JAVA	Controller che permette la rimozione di prodotti dal carrello

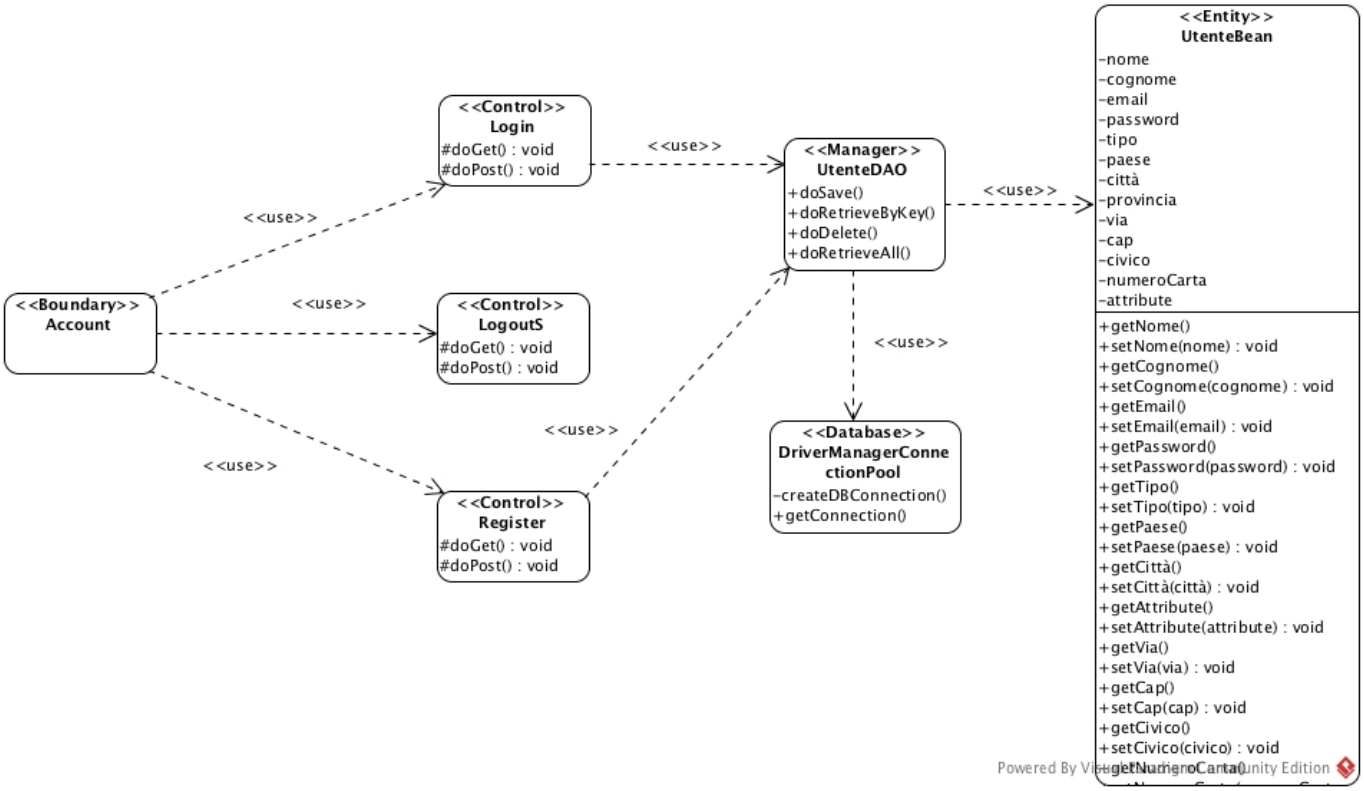
3. Class diagram



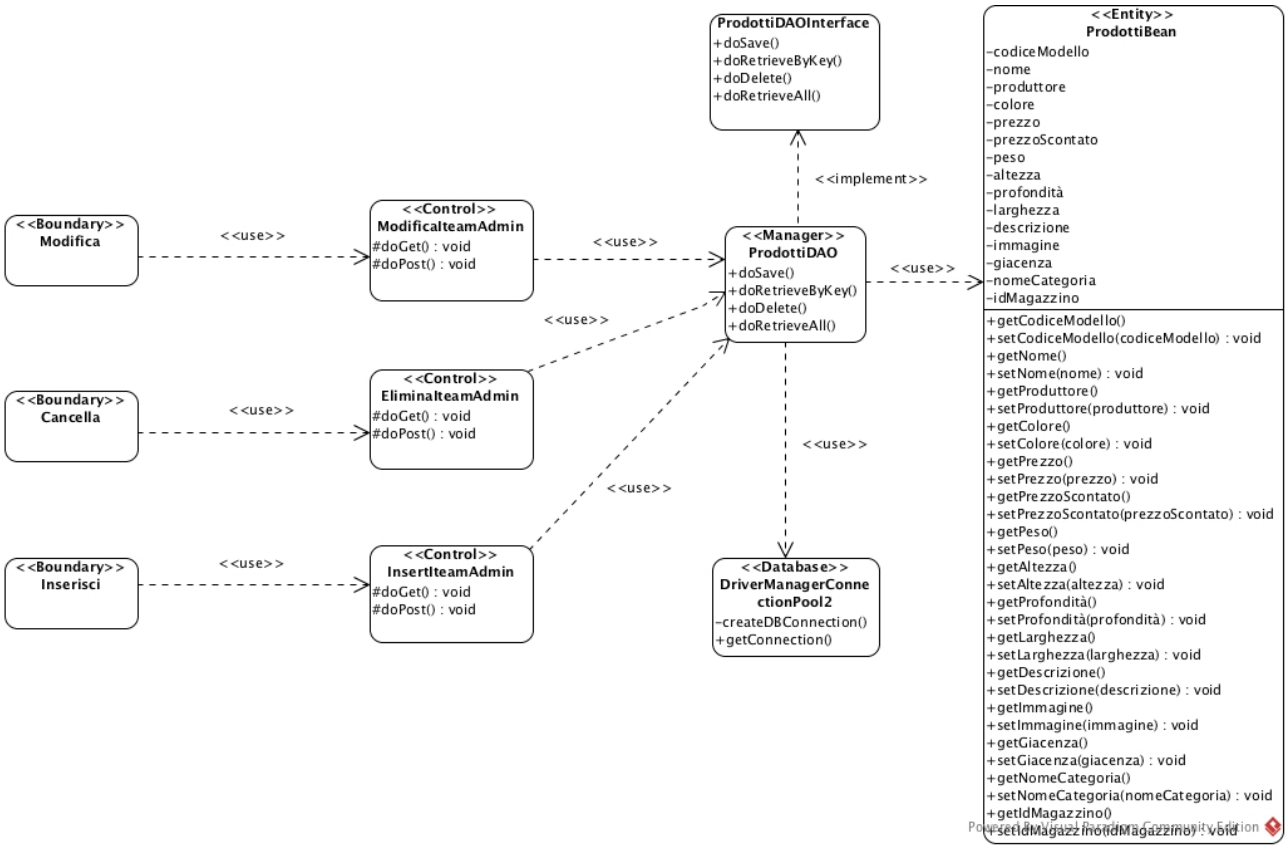
3.1 Control Servlet



3.2 Gestione autenticazione



3.3 Gestione Prodotti



3.4 Gestione Carrello