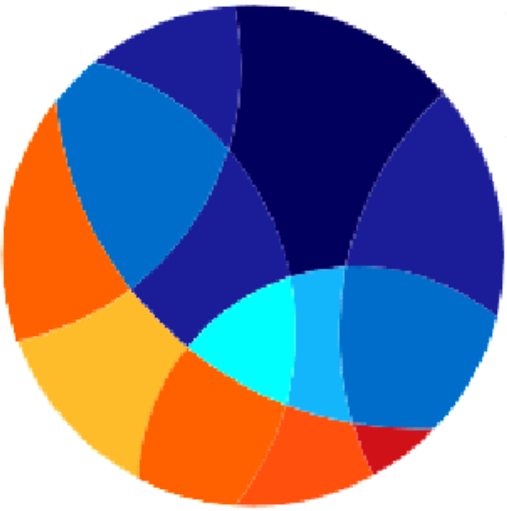
Università degli Studi di Salerno
Corso di Ingegneria del Software

Warehouse Connect
Requirements Analysis Document
Versione 3.0



Data: 05/01/2017

Coordinatore del progetto:

Nome	Matricola
Perrino Francesco	0512102924

Partecipanti:

Nome	Matricola
Perrino Francesco	0512102924
Monaco Simone	0512103206
Santella Pasquale	0512103068

Scritto da:	Monaco Simone/ Francesco Perrino

Revision History

Data	Versione	Descrizione	Autore
07/11/2016	1.0	Prima versione	Monaco Simone/ Francesco Perrino
14/12/2016	2.0	Aggiunta di mock-ups e casi d'uso	Monaco Simone/Francesco Perrino
05/01/2017	3.0	Aggiunta sistema corrente, modifica del capitolo sui modelli (aggiunta di scenari, sequence e class diagram).	Pasquale Santella

Indice

1. INTRODUZIONE

1.1. Descrizione sistema

Il sistema consente la gestione di un database di un magazzino. Tale sistema è basato su una connessione locale ed è implementato su un'interfaccia web che consentirà, a coloro che sono forniti dei dovuti permessi, di compiere particolari operazioni di manutenzione e gestione del database.

1.2. Obiettivo sistema

L'obiettivo del sistema è quello di facilitare la gestione di un magazzino, che finora era gestito manualmente, tramite l'implementazione e l'utilizzo di un software.

1.3. Obiettivi progetto

Gli obiettivi del progetto sono rivolti alla creazione del software il quale deve risultare il più efficiente possibile, sia nell'utilizzo (risultando semplice tuttavia completo) che nell'ottimizzazione delle risorse.

2. SISTEMA CORRENTE

All'accensione del software, l'utente avrà di fronte la pagina principale del programma costituita dal logo del software in alto a sinistra, una barra di ricerca in alto a destra, una tabella a sinistra, con cui cercare i prodotti all'interno del database in base alla categoria, e sulla destra una finestra scorrevole con la lista completa di tutti i prodotti in ordine alfabetico. Quando si andrà a cliccare su un prodotto, si verrà automaticamente portati su una nuova finestra con la scheda di quest'ultimo con tutte le informazioni che si desiderano. In basso a sinistra è presente un pulsante di login dove è possibile immettere username e password e cosi, se riconosciuti, accedere all'area admin. Si presenterà in quel caso una nuova pagina senza lista dei prodotti e con delle modifiche nella tabella a sinistra. Ora la tabella presenta le opzioni di modifica e aggiunta prodotto, oltre alla visualizzazione degli incassi. nella sezione di aggiunta prodotto viene mostrata sulla destra una scheda da compilare con tutte le informazioni necessarie all'aggiunta del prodotto. Se invece si accede all'area di modifica, si avrà di fronte un'altra finestra scorrevole simile a quella presente nell'homepage, ma questa volta sono presenti due pulsanti per la modifica e la rimozione. Quando si decide di eliminare l'oggetto un messaggio di avvenuta rimozione verrà mostrato. Mentre quando si entra nella finestra di modifica vengono presentate le opzioni di modifica di varie caratteristiche dell'oggetto quali il prezzo, la descrizione, etc.
Infine se si clicca sul pulsante "gestione incassi", verrà mostrata una finestra sulla destra con tutte le transazioni avvenute all'interno del negozio.

3. SISTEMA PROPOSTO

3.1. Riassunto

Il sistema consente, generalmente, la gestione di un database. Viene proposto all'utente un software che permette di gestire il lato manageriale del magazzino a cui tale software fa riferimento, permettendo eventualmente a coloro forniti dei dovuti permessi di modificare il database contenente gli articoli immagazzinati. L'utente in questione ha accesso a funzioni di modifica, aggiunta e rimozione di articoli.

3.2. Requisiti funzionali

Il software identifica diversi tipi di attori, quali:

- Uno o più admin, capaci di riempire, svuotare e modificare il database in questione: l'admin si occuperà della gestione diretta del database avendo accesso ad un'interfaccia dedicata;
- Uno o più commessi, capaci di effettuare la ricerca nel database senza però modificarne gli oggetti.

3.3. Requisiti non funzionali

3.3.1. Usabilità

Il software è indirizzato all'utente medio, in questo caso rappresentato da commessi e admin, il quale risulta in grado di utilizzarlo con la massima facilità. Il tutto è permesso da un'interfaccia molto intuitiva.
All'utente utilizzatore del software viene fornita una documentazione riportante la descrizione e le modalità di utilizzo di ciascuna funzionalità dello stesso.

3.3.2. Affidabilità

Il sistema risulta robusto e affidabile, quindi in grado di evitare la perdita e la dispersione di dati. L'utente può riavviare il sistema in caso di malfunzionamento di esso, senza preoccuparsi della perdita di dati. La mole di dati supportabile dal sistema è unicamente limitato dall'hardware della macchina su cui è installato il software, dovendo il database fare affidamento sull'utilizzo di un disco. Alle eccezioni, il sistema reagisce cercando di rimanere stabile senza tuttavia causare cambiamenti e/o problemi al database stesso.

3.3.3. Prestazioni

Il sistema risulta, all'utente, reattivo e veloce. L'utente risulta in grado di fare ciò che deve nel minor tempo e in maniera più efficiente possibile. L'applicazione non ha limiti di utilizzo concorrente da parte di utenti senza permessi di admin, tuttavia solo un admin per volta può accedere e modificare il database.

3.3.4. Compatibilità

L'applicazione supporta cataloghi di tipo .xml. Il sistema è completamente supportato da un server locale e l'interfaccia web può essere acceduta da browser diversi.

3.3.5. Implementazione

Non ci sono vincoli nell'utilizzo legati all'hardware.

3.3.6. Interfaccia

Il sistema non interagisce con sistemi esterni. I dati sono importati ed esportati tramite XML e le immagini possono essere prelevate da siti esterni che fungono da host.

3.3.7. Packaging

L'applicazione è installata su tutti i terminali appartenenti ad una stessa rete, in questo caso la rete è presente all'interno del magazzino, e richiederà solo pochi instanti (al primo avvio) per risultare funzionante.

3.3.8. EULA

L'applicazione è concessa in licenza a chiunque ne richieda una copia. In caso di malfunzionamenti legati all'utilizzo scorretto del software, la responsabilità ricade sull'utilizzatore dello stesso. In caso di malfunzionamenti legati al software e non dipendenti dall'utilizzo scorretto dello stesso (quali perdite di dati, errori imprevisti…) la responsabilità ricade sul manutentore del sistema stesso.

3.4.Modelli

3.4.1. Scenari

Scenario1 : AggiuntaProdotto	
Nome scenario:	**AggiuntaProdotto**
Partecipanti:	James : *Admin*
Flusso degli eventi:	. Selezione dell'opzione per l'inserimento del prodotto . Compilazione degli appositi campi . Sottomissione del form compilato al database

Scenario2 : ModificaProdotto	
Nome scenario:	**ModificaProdotto**
Partecipanti:	James : *Admin*
Flusso degli eventi:	. Selezione dell'opzione per modifica del prodotto . Compilazione degli appositi campi . Sottomissione del form compilato al database

Scenario3 : RimozioneProdotto	
Nome scenario:	**RimozioneProdotto**
Partecipanti:	James : *Admin*
Flusso degli eventi:	. Selezione dell'opzione per la modifica del prodotto . Selezione del pulsante "Rimuovi" . Confermare la rimozione

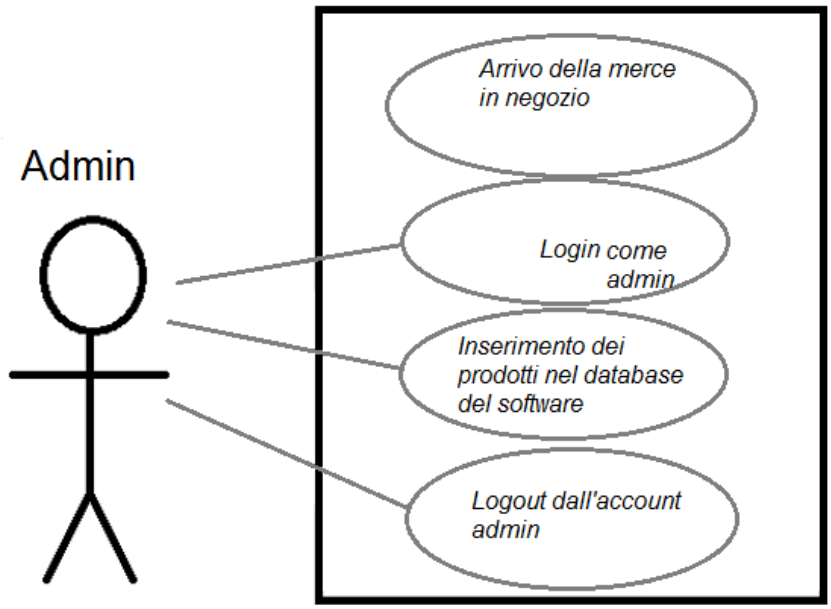
Scenario4 : RicercaProdotto	
Nome scenario:	**RicercaProdotto**
Partecipanti:	Peter : *Commesso*
Flusso degli eventi:	. Selezione dell'opzione per la ricerca nel database . Selezione dei filtri e dei criteri di ricerca . Compilazione degli appositi campi . Sottomissione del form compilato al database

Scenario4 : RicercaProdottoMod	
Nome scenario:	**RicercaProdottoMod**
Partecipanti:	James : *Admin*
Flusso degli eventi:	. Selezione dell'opzione per la ricerca nel database . Selezione dei filtri e dei criteri di ricerca . Compilazione degli appositi campi . Sottomissione del form compilato al database

Scenario4 : LogIn	
Nome scenario:	**LogIn**
Partecipanti:	James : *Admin*
Flusso degli eventi:	. Selezione dell'opzione di login. . Compilazione dei campi di username e password. . Sottomissione del form compilato al database. . Accesso all'area admin, se i dati sono confermati.

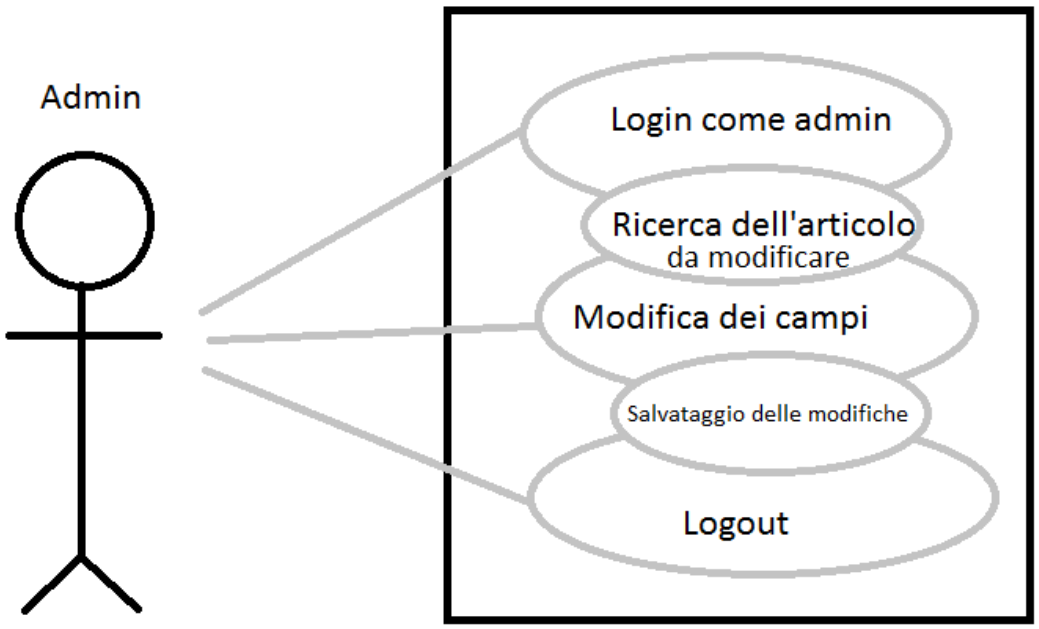
3.4.2. Diagrammi dei casi d'uso

Titolo:	AggiuntaProdotto
Attori partecipanti:	Utente con permessi di admin
Condizioni necessarie:	**.** Il prodotto è disponibile (fisicamente) nel magazzino; **.** Il prodotto non è ancora presente nel database; **.** Lo spazio di archiviazione del database deve poter contenere l'oggetto in questione
Flusso degli eventi:	**.** Selezione dell'opzione per l'inserimento del prodotto **.** Compilazione degli appositi campi **.** Sottomissione del form compilato al database
Condizioni di uscita:	L'oggetto in questione viene aggiunto al database e reso quindi visibile



Titolo:	RimozioneProdotto
Attori partecipanti:	Utente con permessi di admin
Condizioni necessarie:	**.** Il prodotto è disponibile (fisicamente) nel magazzino; **.** Il prodotto è disponibile nel database
Flusso degli eventi:	**.** Selezione dell'opzione per la modifica del prodotto **.** Selezione del pulsante "Rimuovi" **.** Confermare la rimozione
Condizioni di uscita:	L'oggetto in questione viene rimosso dal database e la modifica viene resa visibile

Titolo:	ModificaProdotto
Attori partecipanti:	Utente con permessi di admin
Condizioni necessarie:	**.** Il prodotto è disponibile (fisicamente) nel magazzino; **.** Il prodotto è disponibile nel database **.** Lo spazio di archiviazione del database deve poter supportare le modifiche
Flusso degli eventi:	**.** Selezione dell'opzione per modifica del prodotto **.** Compilazione degli appositi campi **.** Sottomissione del form compilato al database
Condizioni di uscita:	L'oggetto in questione viene modificato al database e reso quindi visibile con le relative modifiche



Titolo:	RicercaProdotto
Attori partecipanti:	**.** Utente generico (commesso)
Condizioni necessarie:	**.** Il prodotto è disponibile (fisicamente) nel magazzino; **.** Il prodotto è disponibile nel database
Flusso degli eventi:	**.** Selezione dell'opzione per la ricerca nel database **.** Selezione dei filtri e dei criteri di ricerca **.** Compilazione degli appositi campi **.** Sottomissione del form compilato al database
Condizioni di uscita:	L'oggetto/Gli oggetti vengono resi visibili in un elenco ordinato secondo i criteri scelti

Titolo:	RicercaProdottoMod
Attori partecipanti:	**.** Utente con permessi di admin
Condizioni necessarie:	**.** Il prodotto è disponibile (fisicamente) nel magazzino; **.** Il prodotto è disponibile nel database
Flusso degli eventi:	**.** Selezione dell'opzione per la ricerca nel database **.** Selezione dei filtri e dei criteri di ricerca **.** Compilazione degli appositi campi **.** Sottomissione del form compilato al database
Condizioni di uscita:	L'oggetto/Gli oggetti vengono resi visibili in un elenco ordinato secondo i criteri scelti

Titolo:	LogIn
Attori partecipanti:	Utente con permessi di admin
Condizioni necessarie:	**.** L'admin è in possesso dei dati per accedere (username e password).
Flusso degli eventi:	**.** Selezione dell'opzione di login. **.** Compilazione dei campi di username e password. **.** Sottomissione del form compilato al database. **.** Accesso all'area admin, se i dati sono confermati.
Condizioni di uscita:	L'admin accede al sistema.

3.4.3. Sequence Diagram

Admin: AggiuntaProdotto

Entity:

- *Admin,* utilizza i suoi privilegi per compiere task speciali.

Boundary:

- *Aggiungi* , bottone presente nella tabella che fa accedere all'area di compilamento dati.
- *Bottone di aggiunta,* da cliccare una volta compilati tutti i campi per poi sottomette il form.
- *Uploader,* bottone per caricare la foto relativa al prodotto che si vuole aggiungere.

Control:

- *AggiungiProdotto.java,* gestisce la funzione ***AggiuntaProdotto.*** Questo oggetto è creato quando l'*Admin* seleziona il bottone *Aggiungi*. Egli in questo modo accede all'area dove devono essere compilati tutti i dati riguardanti l'oggetto. Quando è deciso ad aggiungere il prodotto, con foto allegata caricata grazie all' *Uploader,* si clicca sul *Bottone di aggiunta.* Questo crea un *form* e lo sottomette al *database.*

Admin: ModificaProdotto

Entity:

- *Admin,* utilizza i suoi privilegi per compiere task speciali.

Boundary:

- *Modifica,* bottone da premere per accedere all'area di modifica del prodotto.
- *Bottone di modifica,* bottone presente ad ognuno dei prodotti presenti nella lista. L'admin deve cliccare quello del prodotto che desidera modificare.
- *Bottone di conferma,* da premere nel caso in cui si vuole salvare le modificare al prodotto selezionare

Control:

- *ModificaProdotto.java,* gestisce la funzione ***ModificaProdotto.*** Questo oggetto è creato quando l'*Admin* seleziona il bottone *Modifica*. In questo modo, l'admin accedere alla nuova finestra in cui è presente la lista degli elementi presenti nel database con ognuno al suo fianco due bottoni : Bottone di rimozione e *Bottone di modifica.* Una volta cliccato sul *Bottone di modifica* egli accede in una nuova finestra in cui è possibile modificare liberamente alcune features del prodotto quali prezzo, descrizione, etc…
  Quando si è soddisfatti delle modifiche si clicca il *Bottone di conferma* per immettere il *form* al *database.*

Admin: RimozioneProdotto

Entity:

- *Admin,* grazie ai suoi privilegi accede all'area admin.

Boundary:

- *Modifica,* bottone presente nella tabelle per accedere all'area di modifica dei prodotti.
- *Bottone di rimozione,* bottone presente al fianco di ogni prodotto. Quando viene cliccato il prodotto a cui corrisponde, esso viene cancellato dal database.

Control:

- *servletTramite,* gestisce la funzione ***RimozioneProdotto.*** Questo oggetto è creato quando l'*Admin* seleziona il *Bottone di rimozione*.

Utente: RicercaProdotto

Entity:

- *Utente*, utilizza il sistema per cercare un prodotto all'interno del database.

Boundary:

- *Cerca*, bottone utilizzato per accedere alla finestra di ricerca**.**
- *Bottone di ricerca*, utilizzato una volta che sono stati riempiti gli appositi campi per cominciare la ricerca all'interno del database.

Control:

- *ricerca e ricercanome*, gestisce la funzione ***RicercaProdotto.*** Questo oggetto è creato quando l'*Utente* seleziona il bottone *Cerca*. In questo modo egli accede a una nuova finestra di pinup dove gli è possibile scegliere tra due diversi tipi di ricerca : ricerca per ID, con *ricerca* come control, e ricerca per nome, tipologia e marca, che ha come control *ricercanome*. Quando poi si clicca il *Bottone di ricerca*, viene inviato un *form* al *database.* Vengono infine mostrati i risultati della ricerca.

Utente: RicercaProdottoMod

Entity:

- *Admin*, utilizza il sistema per cercare un prodotto all'interno del database.

Boundary:

- *Cerca*, bottone utilizzato per accedere alla finestra di ricerca**.**
- *Bottone di ricerca*, utilizzato una volta che sono stati riempiti gli appositi campi per cominciare la ricerca all'interno del database.

Control:

- *ricercamod e ricercanomemod*, gestisce la funzione ***RicercaProdottoMod.*** Questo oggetto è creato quando l'*Admin* seleziona il bottone *Cerca*. In questo modo egli accede a una nuova finestra di pinup dove gli è possibile scegliere tra due diversi tipi di ricerca : ricerca per ID, con *ricercamod* come control, e ricerca per nome, tipologia e marca, che ha come control *ricercanomemod*. Quando poi si clicca il *Bottone di ricerca*, viene inviato un *form* al *database.* Vengono infine mostrati i risultati della ricerca.

Admin: LogIn

Entity:

- *Admin,* è in grado di poter entrare nell'area admin grazie ai dati che gli sono stati forniti

Boundary:

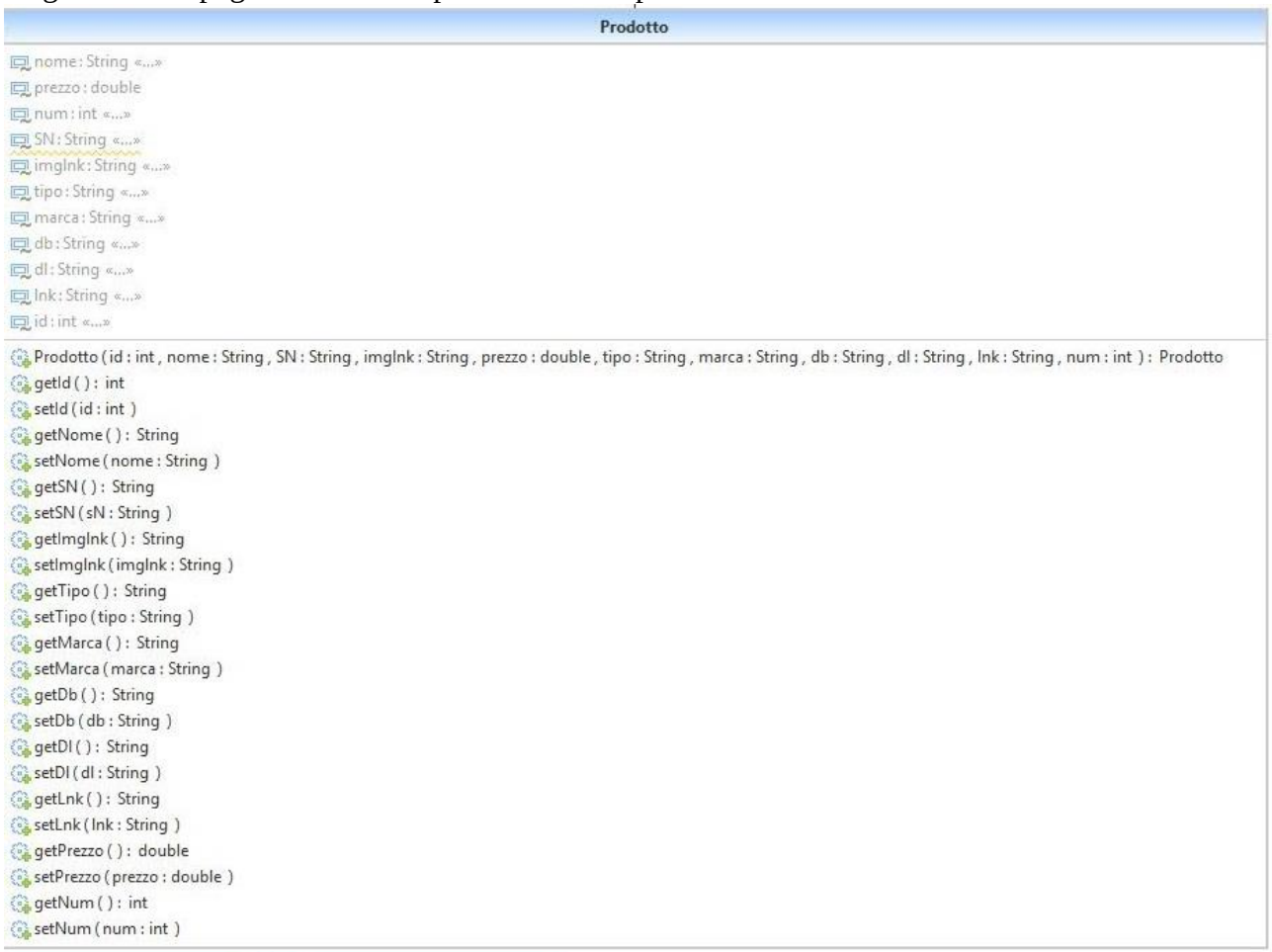
- *Area Gestione,* link presenta in basso a sinistra nella homepage dell'utente che fa apparire la finestra di login.
- *Accedi*, bottone presente nella finestra di login che, una volta compilati i campi di username e password correttamente, fa accedere all'area admin del programma.

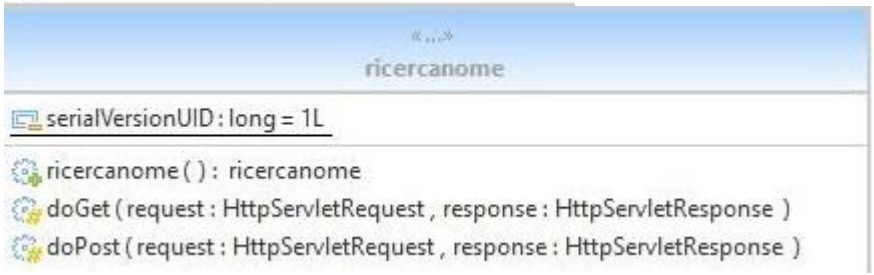
Control:

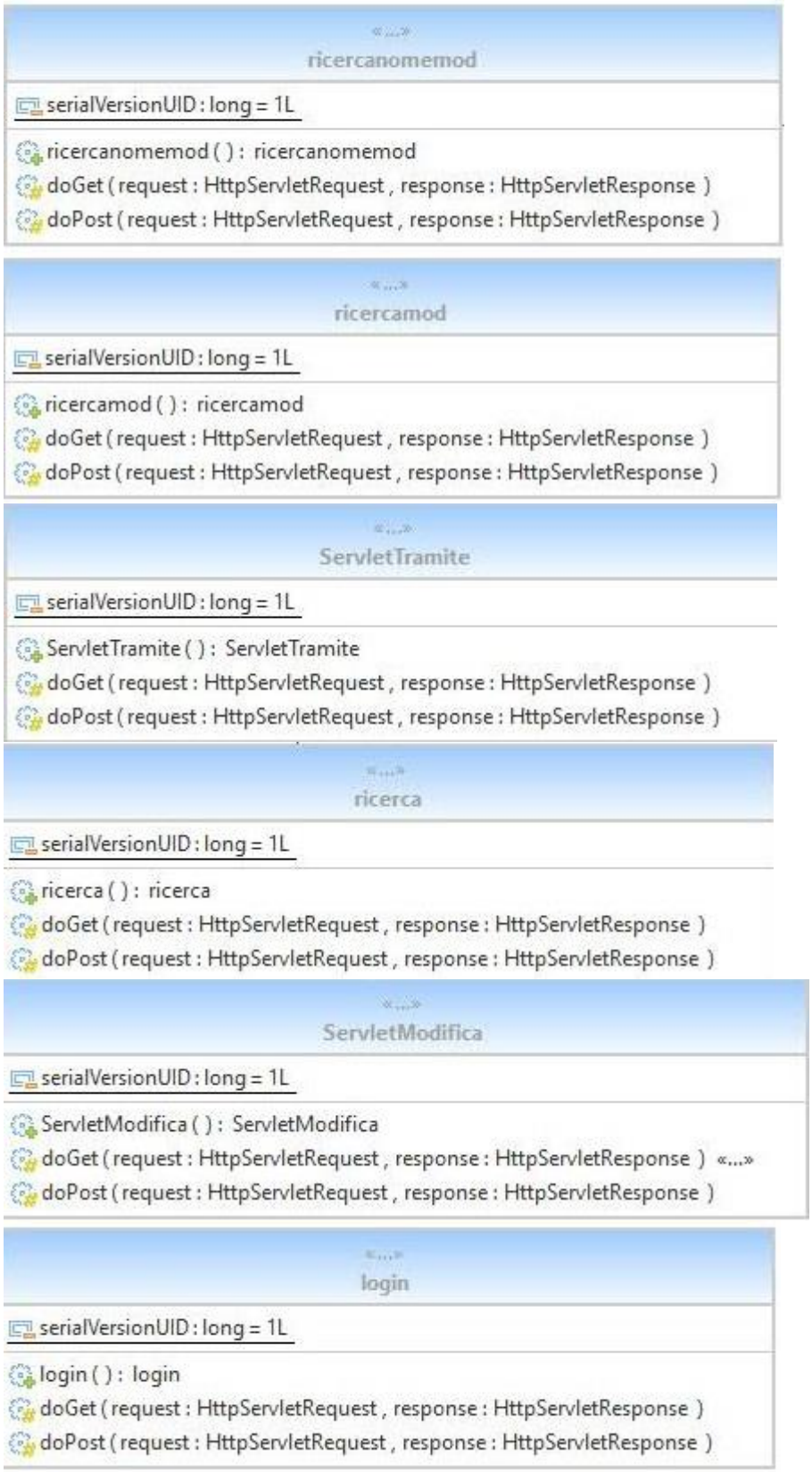
- *login*, gestisce la funzione ***LogIn.*** Questo oggetto è creato quando l'*Admin* seleziona il link *Area* Gestione. Una volta aperta la pagina ed aver correttamente compilati i dati, cliccando il tasto *Accedi* viene creato un *form* al *database*. Se i dati sono corretti egli accederà all'area admin.

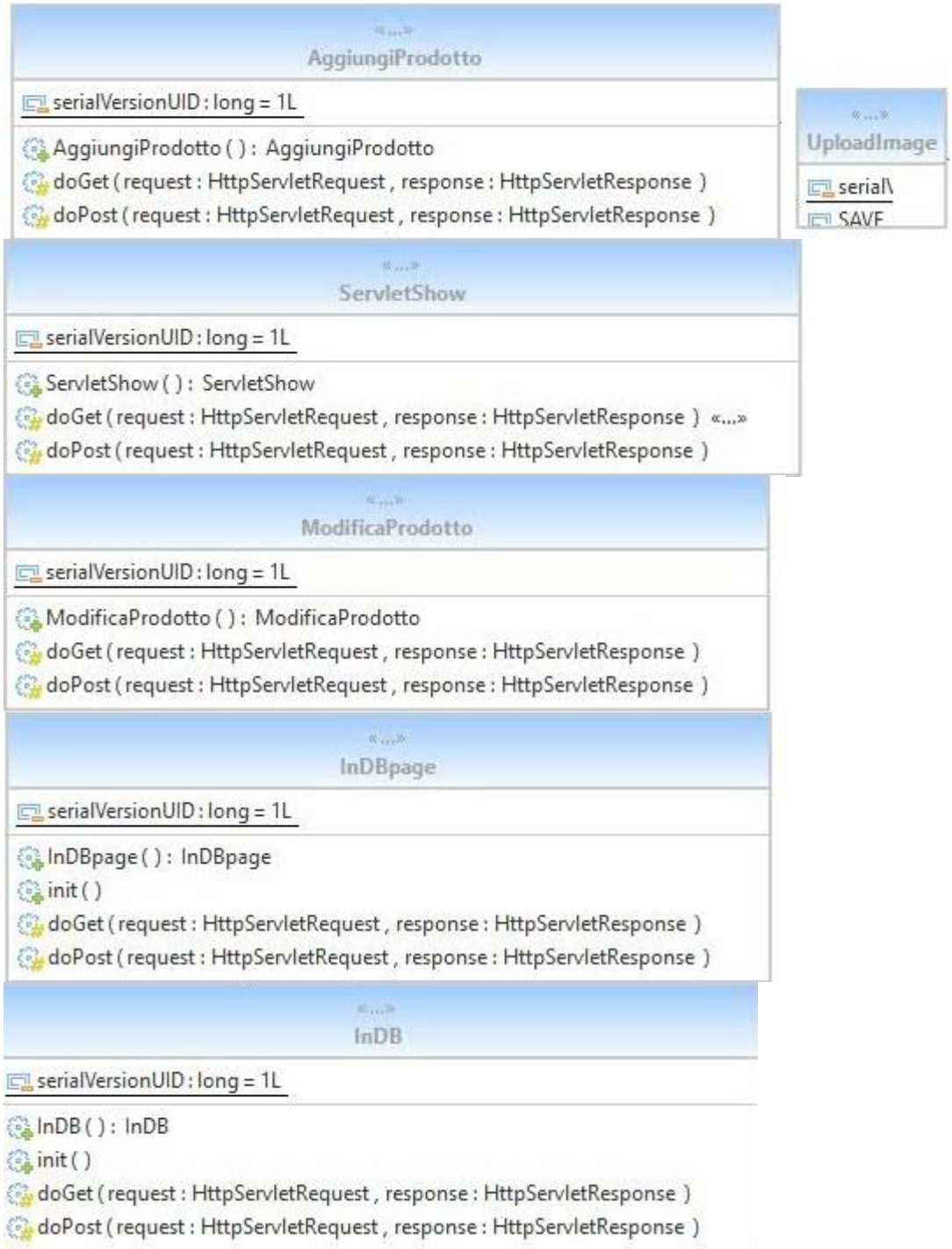
3.4.4. Class diagram

Di seguito riportiamo tutte le classi POJO determinate dall'analisi degli use case e dei sequence diagram. Nelle pagine successive presentiamo un primo modello raffinato



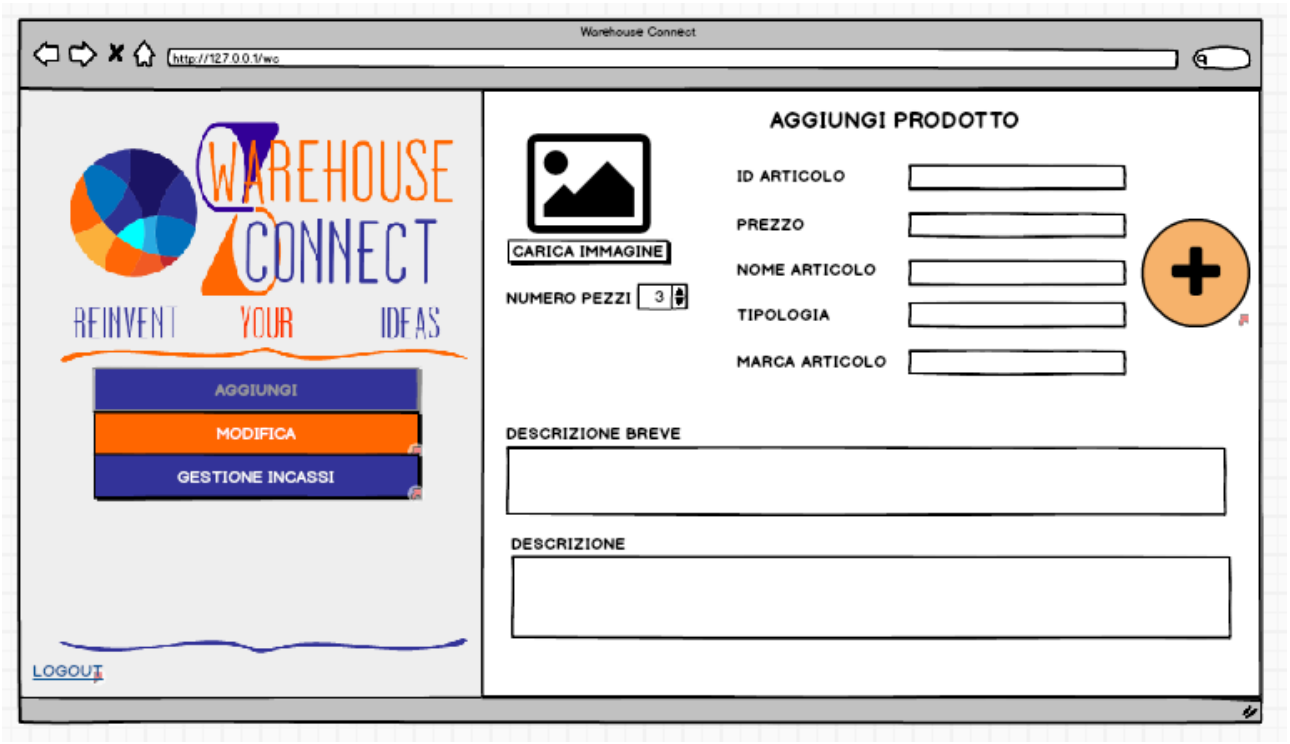



«...»
ricercanomemod
serialVersionUID : long = 1L
ricercanomemod ( ) : ricercanomemod
doGet (request : HttpServletRequest , response : HttpServletResponse)
doPost (request : HttpServletRequest , response : HttpServletResponse)
«...»
ricercamod
serialVersionUID : long = 1L
ricercamod ( ) : ricercamod
doGet (request : HttpServletRequest , response : HttpServletResponse)
doPost (request : HttpServletRequest , response : HttpServletResponse)
«...»
ServletTramite
serialVersionUID : long = 1L
ServletTramite ( ) : ServletTramite
doGet (request : HttpServletRequest , response : HttpServletResponse)
doPost (request : HttpServletRequest , response : HttpServletResponse)
«...»
ricerca
serialVersionUID : long = 1L
ricerca ( ) : ricerca
doGet (request : HttpServletRequest , response : HttpServletResponse)
doPost (request : HttpServletRequest , response : HttpServletResponse)
«...»
ServletModifica
serialVersionUID : long = 1L
ServletModifica ( ) : ServletModifica
doGet (request : HttpServletRequest , response : HttpServletResponse) «...»
doPost (request : HttpServletRequest , response : HttpServletResponse)
«...»
login
serialVersionUID : long = 1L
login ( ) : login
doGet (request : HttpServletRequest , response : HttpServletResponse)
doPost (request : HttpServletRequest , response : HttpServletResponse)

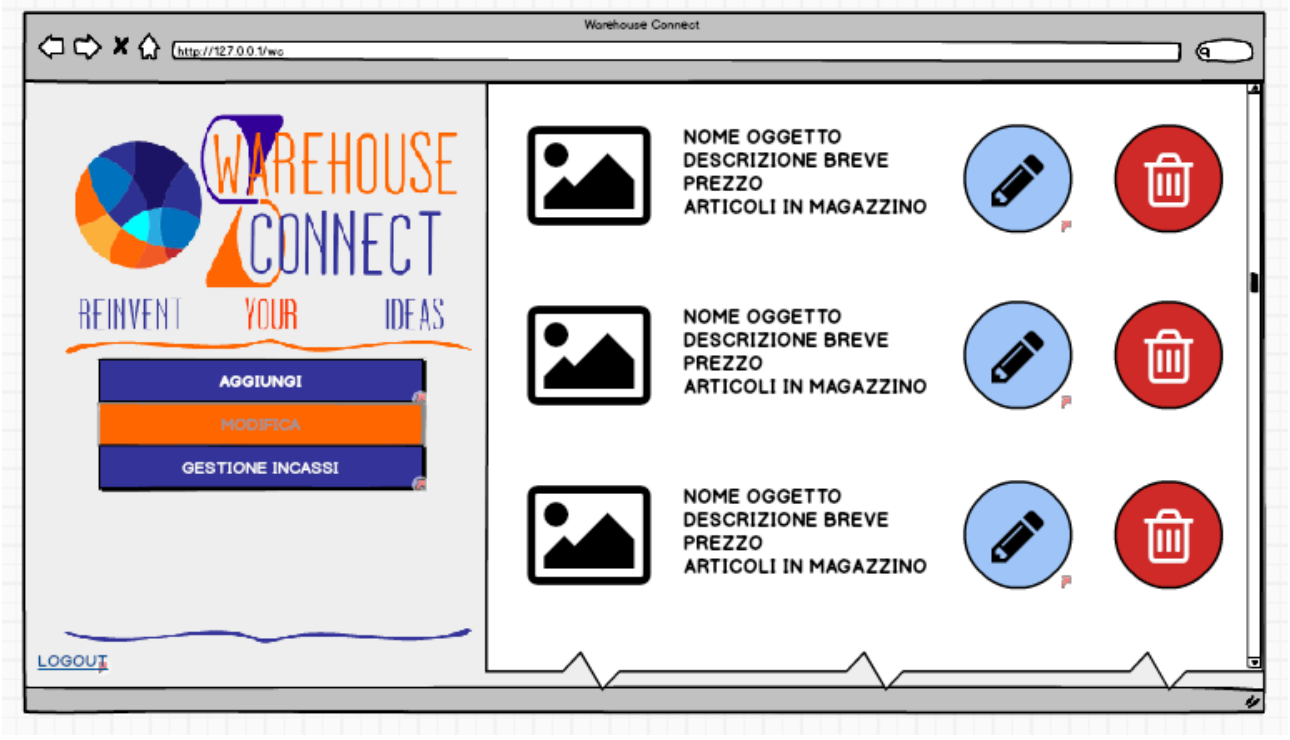

«...»
AggiungiProdotto
serialVersionUID : long = 1L
AggiungiProdotto ( ) : AggiungiProdotto
doGet (request : HttpServletRequest , response : HttpServletResponse)
doPost (request : HttpServletRequest , response : HttpServletResponse)
«...»
UploadImage
serial\
SAVE
«...»
ServletShow
serialVersionUID : long = 1L
ServletShow ( ) : ServletShow
doGet (request : HttpServletRequest , response : HttpServletResponse) «...»
doPost (request : HttpServletRequest , response : HttpServletResponse)
«...»
ModificaProdotto
serialVersionUID : long = 1L
ModificaProdotto ( ) : ModificaProdotto
doGet (request : HttpServletRequest , response : HttpServletResponse)
doPost (request : HttpServletRequest , response : HttpServletResponse)
«...»
InDBpage
serialVersionUID : long = 1L
InDBpage ( ) : InDBpage
init ( )
doGet (request : HttpServletRequest , response : HttpServletResponse)
doPost (request : HttpServletRequest , response : HttpServletResponse)
«...»
InDB
serialVersionUID : long = 1L
InDB ( ) : InDB
init ( )
doGet (request : HttpServletRequest , response : HttpServletResponse)
doPost (request : HttpServletRequest , response : HttpServletResponse)

3.4.5. Schema UI e mock-ups

La schermata di ricerca accessibile dalla finestra principale



La schermata di aggiunta di un prodotto accessibile dalla schermata dell'admin



La schermata di modifica