



Università degli Studi di Salerno

Corso di Ingegneria del Software

e-comix

ODD
Versione 1.2



Data: 30/11/2017

Progetto: e-comix	Versione: 1.2
Documento: ODD	Data: 30/11/2017

Partecipanti:

Nome	Matricola
Gerardo De Rosa	0512103762
Pasquale Nappo	0512103046
Gerardo Ragosta	0512101794

Scritto da:	Gerardo De Rosa, Pasquale Nappo, Gerardo Ragosta
--------------------	--

Revision History

Data	Versione	Descrizione	Autore
30/11/2017	1.0	Stesura Introduzione e Pacchetti	Gerardo De Rosa
02/12/2017	1.1	Stesura Interfacce delle Classi e Glossario	Gerardo De Rosa
10/12/2017	1.2	Revisione generale dell'intero documento	Gerardo Ragosta

Progetto: e-comix	Versione: 1.2
Documento: ODD	Data: 30/11/2017

Indice

1. Introduzione	4
1.1 Object design trade-off	4
1.1.1 Comprensibilità vs costi	4
1.1.2 Prestazioni vs Costi	4
1.1.3 Costi vs Mantenimento	4
1.1.4 Interfaccia vs Easy-Use	4
1.1.5 Memoria vs efficienza	4
1.1.6 Sicurezza vs Costi	4
1.1.7 Interfaccia vs Tempo di Risposta	5
1.2 Linee guida per la documentazione dell'interfaccia	5
1.3 Definizioni, acronimi e abbreviazioni	6
1.4 Riferimenti	6
2. Pacchetti	7
2.1 PK - Pacchetti generali	7
2.2 PK - Pacchetto Interfacce Generale	8
2.2.1 PK - Pacchetto Interfacce Gestione Utente (Cliente)	9
2.2.2 PK - Pacchetto Interfacce Gestione Utente (Gestore)	9
2.2.3 PK - Pacchetto Interfaccia Gestione Ordini	10
2.2.4 PK - Pacchetto Interfaccia Gestione Prodotti	10
2.3 PK - Entità	11
2.4 PK - Package Gestione sottoinsiemi	16
2.5 PK - Storage	19
3. Interfacce delle classi	20
3.1 CD GestioneUtente	20
3.2 CD GestioneProdotti (Generale)	21
3.2.1 CD GestioneProdotti (Confronto)	22
3.2.2 CD GestioneProdotti (Carrello + Wishlist)	23
3.2.3 CD GestioneProdotti (Visualizza + Ricerca)	24
3.2.4 CD GestioneProdotti (Eliminazione/Modifica/Inserimento)	25
3.3 CD GestioneOrdini	26
4. Glossario	27

Progetto: e-comix	Versione: 1.2
Documento: ODD	Data: 30/11/2017

1. Introduzione

1.1 Object design trade-off

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.

1.1.1 Comprensibilità vs costi

Si preferisce aggiungere costi relativi alle ore/uomo dedicate per la documentazione, al fine di rendere il codice comprensibile sia alle persone non coinvolte nel progetto che alle persone coinvolte che non hanno lavorato a quella parte in particolare.

1.1.2 Prestazioni vs Costi

Considerando che il budget a nostra disposizione non è eccessivo, il sistema sarà sviluppato utilizzando componenti free e open source, non saranno garantite alte prestazioni, ma queste saranno soddisfacenti per il normale utilizzo.

1.1.3 Costi vs Mantenimento

Il sistema può essere facilmente modificato ed implementato con nuove funzioni e corretto in presenza di errori, grazie all'uso di materiale open source.

1.1.4 Interfaccia vs Easy-Use

L'interfaccia, grazie ad un'impostazione semplice e intuitiva, permette facile utilizzo (Easy-use) delle principali funzionalità del sistema anche per gli utenti meno esperti.

1.1.5 Memoria vs efficienza

Dato l'elevato carico di utenti che possono accedere al sistema, e dato che il sistema dovrà supportare un elevato numero di query, per non penalizzare le performance del sistema, verrà utilizzato un meccanismo di caching che permette la memorizzazione dei dati con accesso più frequente. Ciò sarà implementato attraverso l'utilizzo di cookie.

1.1.6 Sicurezza vs Costi

Dato il budget ridotto, non saranno utilizzate componenti esterne che

	Ingegneria del Software	Pagina 4 di 27
--	-------------------------	----------------

Progetto: e-comix	Versione: 1.2
Documento: ODD	Data: 30/11/2017

garantiscano la massima sicurezza sui dati, ma verrà utilizzata una componente all'interno del linguaggio java che permetta un grado di protezione soddisfacente.

1.1.7 Interfaccia vs Tempo di Risposta

Il tempo di risposta tra server e interfaccia è più che sufficiente a soddisfare le richieste da parte dell'utente.

1.2 Linee guida per la documentazione dell'interfaccia

Per rendere il codice più estensibile e manutenibile, prima dell'implementazione della logica del sistema, è opportuno sottomettere le regole di implementazione, in modo che eventuali correzioni nella logica dell'applicazione possano essere apportate prima di imbattersi nella sintassi degli strumenti scelti.

Dichiarazioni

Posizionare le dichiarazioni all'inizio del blocco del codice. Non dichiarare le variabili al loro primo uso, può portare incomprensioni verso il programmatore e rendere la manutenibilità del codice più complessa e di difficile comprensione.

Indentazione

L'indentazione deve essere effettuata con un TAB e a prescindere del linguaggio usato per la produzione del codice, ogni istruzione deve essere opportunamente indentata.

Es.

```
<html>
<head>
</head>
<body>
</body>
</html>
```

Deve essere sostituito da:

```
<html>
    <head>
    </head>
    <body>
```

Progetto: e-comix	Versione: 1.2
Documento: ODD	Data: 30/11/2017

```

        </body>
</html>

```

Parentesi

A prescindere dalle istruzioni un IF o un ciclo FOR e WHILE, è necessario laddove ci fosse anche una sola istruzione, riportare il blocco di istruzioni tra parentesi graffe.

1.3 Definizioni, acronimi e abbreviazioni

ODD: Object Design Document.

PK: Pacchetti.

CD: Class Diagram.

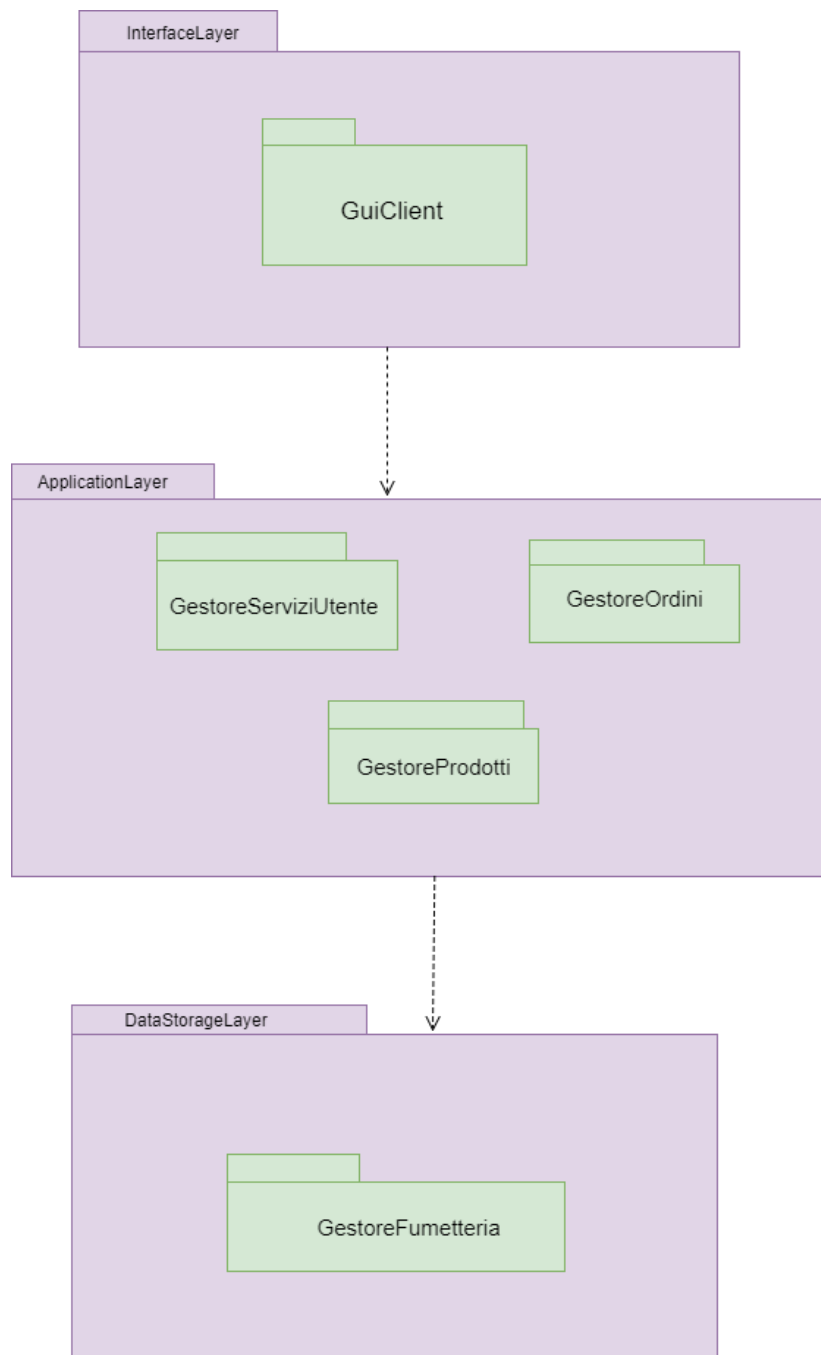
1.4 Riferimenti

- RAD_e-comix
- SDD e-comix
- Bruegge, Dutoit, Object-Oriented Software Engineering.
- Sommerville, Software Engineering-Addison Wesley.

Progetto: e-comix	Versione: 1.2
Documento: ODD	Data: 30/11/2017

2. Pacchetti

2.1 PK - Pacchetti generali



Il diagramma descrive la natura three-layer dell'applicazione mostrandone i tre package principali:

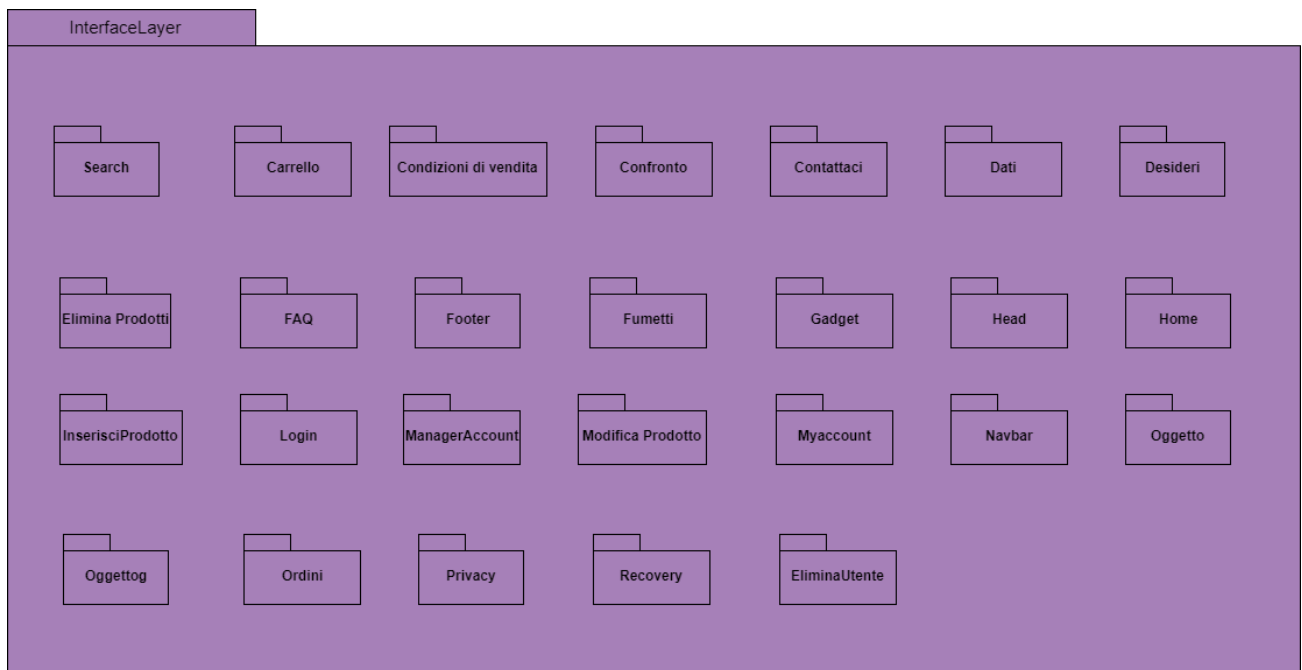
Progetto: e-comix	Versione: 1.2
Documento: ODD	Data: 30/11/2017

InterfaceLayer: GUIClient, indica i sottosistemi che contengono tutti gli oggetti boundary;

ApplicationLogicLayer: contiene i quattro sottosistemi individuati (**GestioneServiziUtente**, **GestoreOrdini**, **GestoreProdotti**) ;

DataStorageLayer: GestoreFumetteria, sottosistema che ha il compito di effettuare operazioni verso il database.

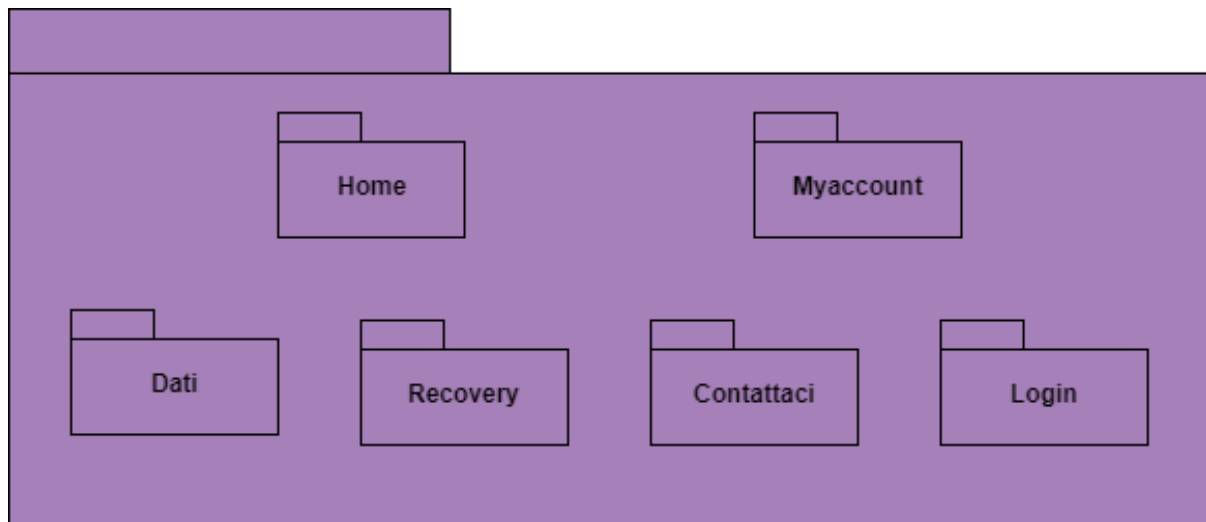
2.2 PK - Pacchetto Interfacce Generale



Il diagramma descrive le interfacce delle varie sezioni del sistema: Gestione Utente, Gestione Ordini, Gestione Prodotto.

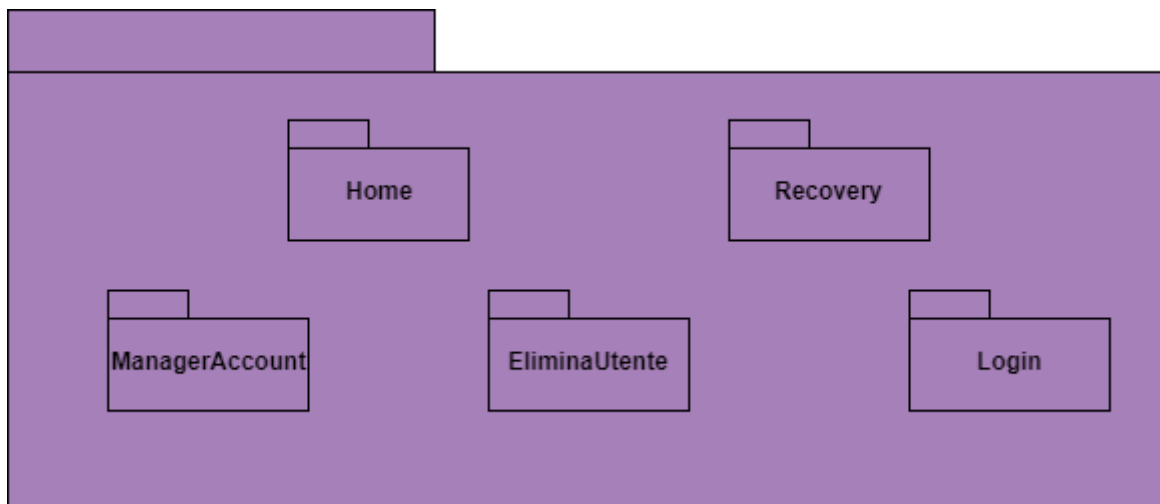
Progetto: e-comix	Versione: 1.2
Documento: ODD	Data: 30/11/2017

2.2.1 PK - Pacchetto Interfacce Gestione Utente (Cliente)



Il diagramma descrive le interfacce del sottosistema "Gestione Utente" ed in particolare le View per il cliente.

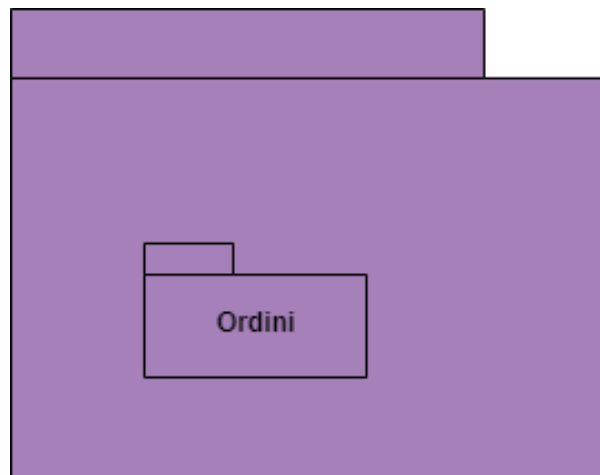
2.2.2 PK - Pacchetto Interfacce Gestione Utente (Gestore)



Il diagramma descrive le interfacce del sottosistema "Gestione Utente" ed in particolare le View del Gestore del sito.

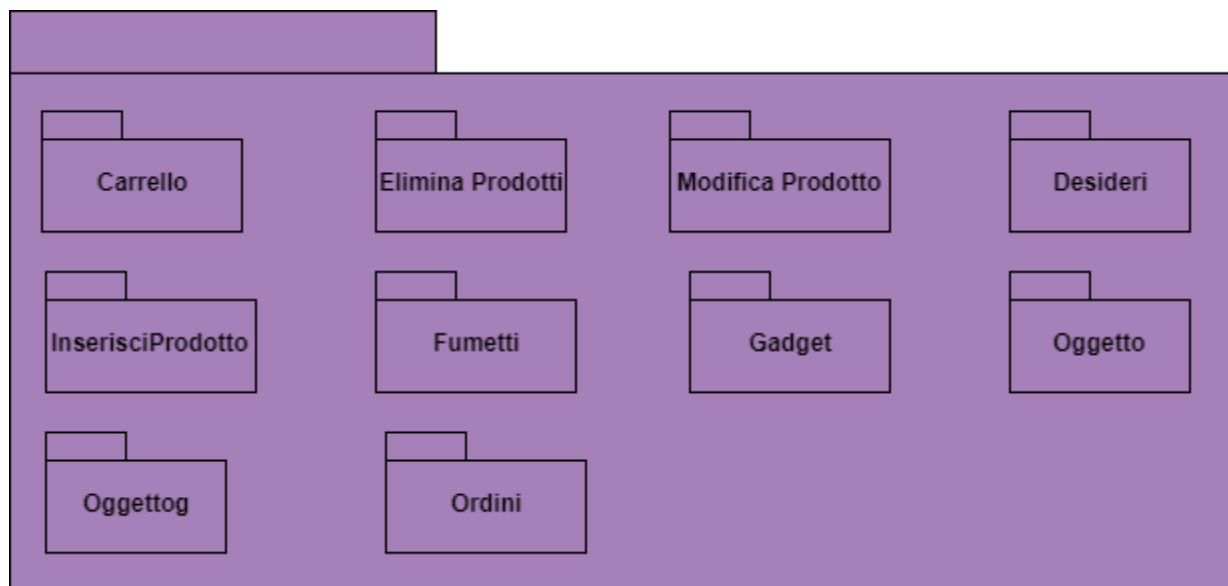
Progetto: e-comix	Versione: 1.2
Documento: ODD	Data: 30/11/2017

2.2.3 PK - Pacchetto Interfaccia Gestione Ordini



Il diagramma descrive le interfacce del sottosistema "Gestione Ordini".

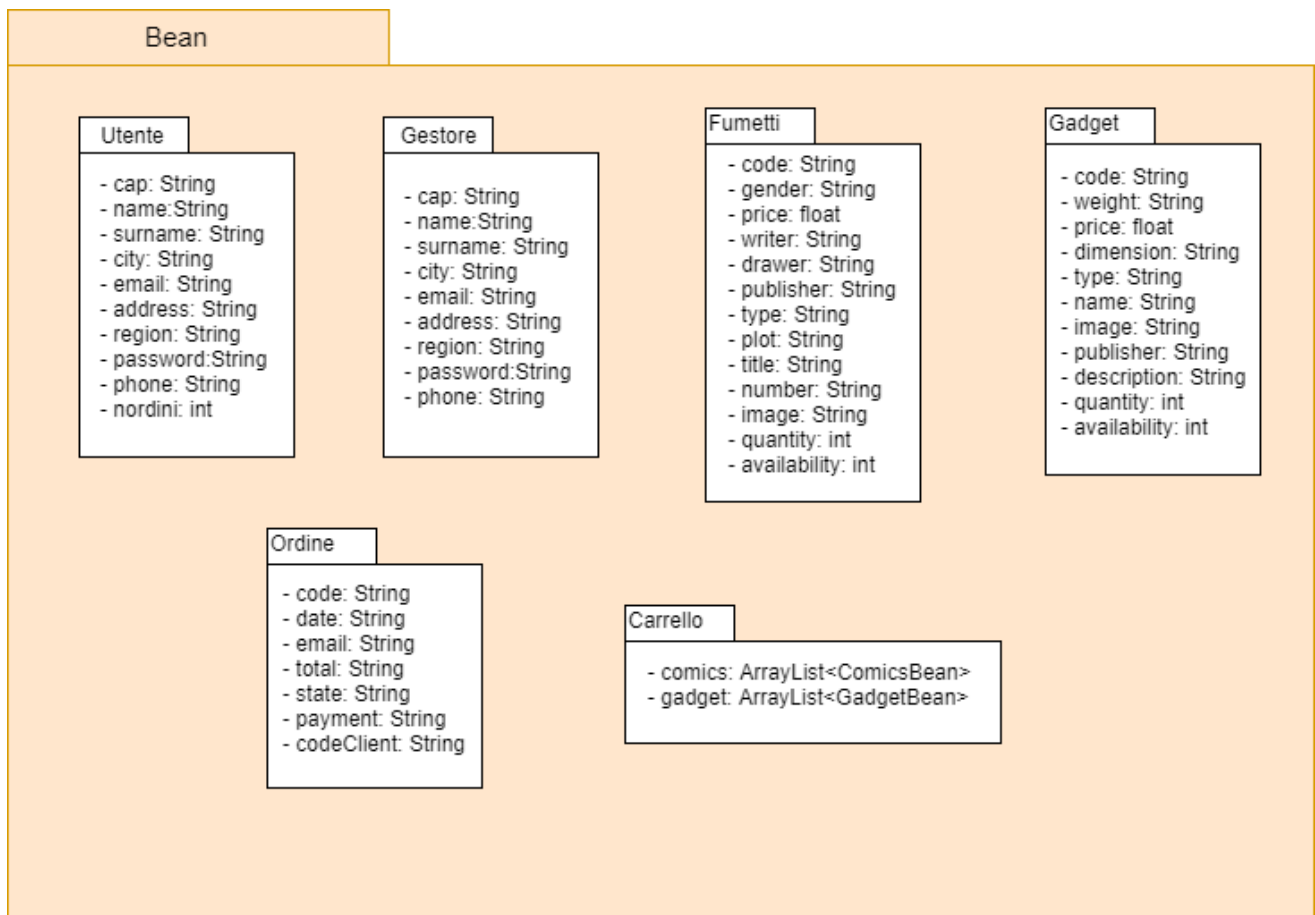
2.2.4 PK - Pacchetto Interfaccia Gestione Prodotti



Il diagramma descrive le interfacce del sottosistema "Gestione Prodotti".

Progetto: e-comix	Versione: 1.2
Documento: ODD	Data: 30/11/2017

2.3 PK - Entità



I “bean” ovvero le entity del nostro sistema.
Per ogni entità sono definite le variabili e il tipo di quest’ultime.

Utente

Nome	Utente
Descrizione	Questa classe rappresenta l’utente registrato al sito.
Signature dei Metodi	+ getCap() + setCap(String cap) + getName() + setName(String name) + getSurname() + setSurname(String surname) + getCity() + setCity(String city) + getAddress() + setAddress(String address) + getEmail()

Progetto: e-comix	Versione: 1.2
Documento: ODD	Data: 30/11/2017

	+ setEmail(String email) + getRegion() + setRegion(String region) + getPassword() + setPassword(String password) + getPhone() + setPhone(String phone) + getNordini() + setNordini(int nordini)
Precondizione	context: Utente:: setEmail(String email) pre: Email non deve avere altre corrispondenze nel database.
Post-Condizione	
Invariante	

Gestore

Nome	Gestore
Descrizione	Questa classe rappresenta l'utente registrato al sito.
Signature dei Metodi	+ getCap() + setCap(String cap) + getName() + setName(String name) + getSurname() + setSurname(String surname) + getCity() + setCity(String city) + getAddress() + setAddress(String address) + getEmail() + setEmail(String email) + getRegion() + setRegion(String region) + getPassword() + setPassword(String password) + getPhone() + setPhone(String phone)
Precondizione	context: Utente:: setEmail(String email) pre: Email non deve avere altre corrispondenze nel database.
Post-Condizione	
Invariante	

Progetto: e-comix	Versione: 1.2
Documento: ODD	Data: 30/11/2017

Ordine

Nome	Ordine
Descrizione	Questa classe rappresenta gli ordini effettuati dagli utenti.
Signature dei Metodi	+ getcode() + setcode(String code) + getemail() + setemail(String email) + getdate() + setdate(String date) + gettotal() + setttotal(String total) + getstate() + setstate(String state) + getpayment() + setpayment(String payment) + getcodeClient() + setcodeClient(String codeClient)
Precondizione	context: Ordine:: setcode(String code) pre: Code non deve avere altre corrispondenze nel database.
Post-Condizione	
Invariante	

Carrello

Nome	Carrello
Descrizione	Questa classe rappresenta il carrello dove gli utenti aggiungono prodotti.
Signature dei Metodi	+ addComic(ComicsBean comic) + addGadget(GadgetBean gadget) + deleteComic(ComicsBean comic) + deleteGadget(GadgetBean gadget)
Precondizione	
Post-Condizione	
Invariante	

Fumetto

Nome	Fumetto
Descrizione	Questa classe rappresenta la tipologia di prodotto fumetto.

Progetto: e-comix	Versione: 1.2
Documento: ODD	Data: 30/11/2017

Signature dei Metodi	+ getavailability() + setavailability(int availability) + getcode() + setcode(String code) + getquantity() + setquantity(int quantity) + getgender() + setgender(String gender) + getprice() + setprice(float price) + getwriter() + setwriter(String writer) + getdrawer() + setdrawer(String drawer) + getpublisher() + setpublisher(String publisher) + gettype() + settype(String type) + getplot() + setplot(String plot) + gettitle() + setttitle(String title) + getnumber() + setnumber(String number) + getimage() + setimage(String image)
Precondizione	context: Fumetto :: setcode(String code) pre: Code non deve avere altre corrispondenze nel database.
Post-Condizione	
Invariante	

Gadget

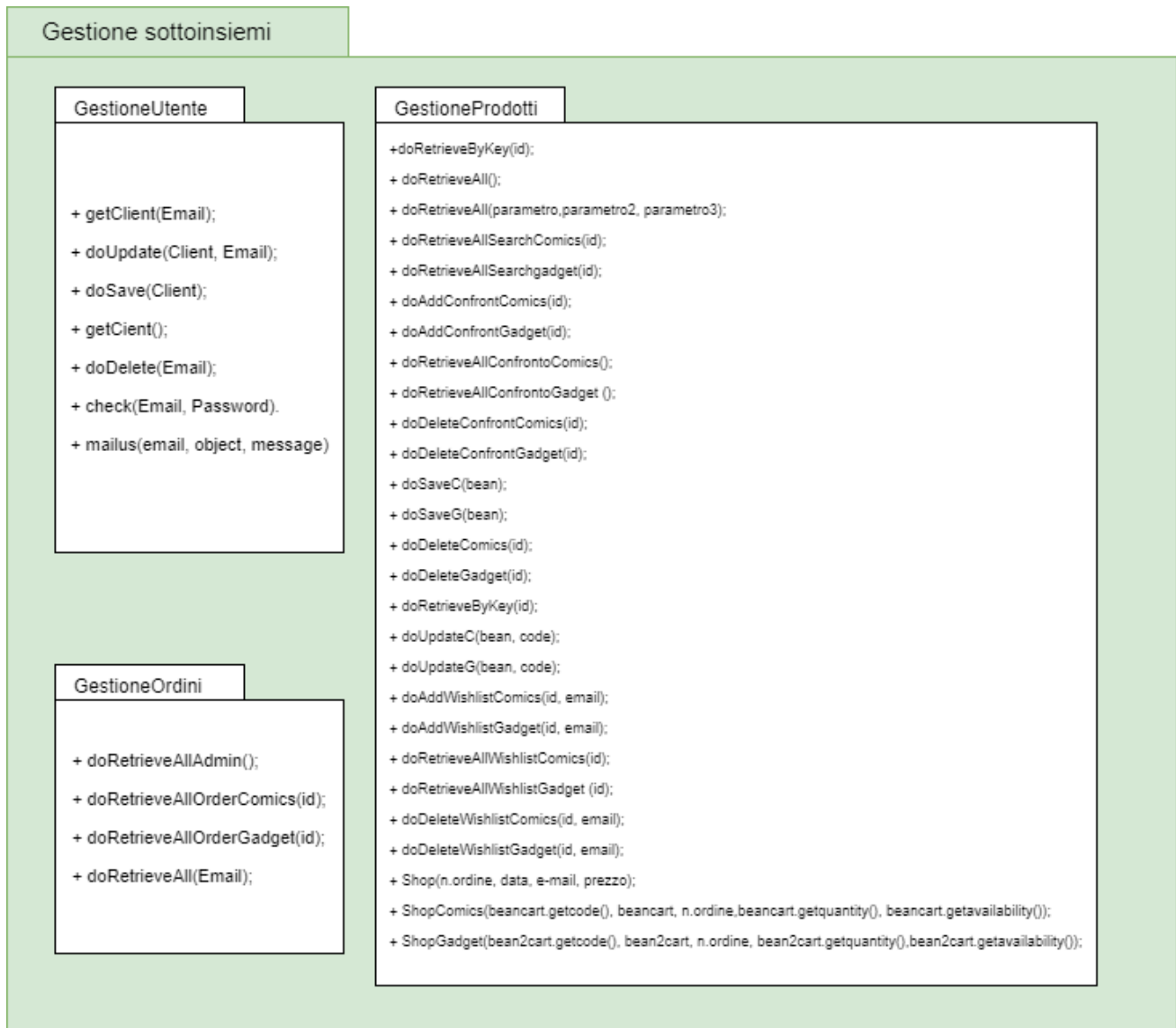
Nome	Gadget
Descrizione	Questa classe rappresenta la tipologia di prodotto gadget.
Signature dei Metodi	+ getavailability() + setavailability(int availability) + getcode() + setcode(String code) + getquantity() + setquantity(int quantity) + getpublisher() + setpublisher(String publisher)

Progetto: e-comix	Versione: 1.2
Documento: ODD	Data: 30/11/2017

	+ getdescription() + setdescription(String description) + getweight() + setweight(String weight) + getprice() + setprice(float price) + getdimension() + setdimension(String dimension) + gettype() + settype(String type) + getname() + setname(String name) + getimage() + setimage(String image)
Precondizione	context: Gadget:: setcode(String code) pre: Code non deve avere altre corrispondenze nel database.
Post-Condizione	
Invariante	

Progetto: e-comix	Versione: 1.2
Documento: ODD	Data: 30/11/2017

2.4 PK - Package Gestione sottoinsiemi



I package in “Gestione sottosistemi” fanno parte dello strato applicativo e si focalizzano sui pacchetti che contengono le classi che si occuperanno di implementare la logica del sistema (oggetti control presenti nel RAD).

GestioneProdotti

Nome	GestioneProdotti
Descrizione	Questo pacchetto fornisce varie funzionalità per permettere la gestione dei prodotti.
Signature dei Metodi	- doRetrieveByKey(id); - doRetrieveAll(); - doRetrieveAll(parametro, parametro2,

Progetto: e-comix	Versione: 1.2
Documento: ODD	Data: 30/11/2017

	parametro3); - doAddConfrontComics(id); - doAddConfrontGadget(id); - doRetrieveAllConfrontoComics(); - doRetrieveAllConfrontoGadget (); - doDeleteConfrontComics(id); - doDeleteConfrontGadget(id); - doAddWishlistComics(id, email); - doAddWishlistGadget(id, email); - doRetrieveAllWishlistComics(id); - doRetrieveAllWishlistGadget (id); - doDeleteWishlistComics(id, email); - doDeleteWishlistGadget(id, email); - Shop(n.ordine, data, e-mail, prezzo); - ShopComics(bean2cart.getcode(), bean2cart, n.ordine, bean2cart.getquantity(), bean2cart.getavailability()); - ShopGadget(bean2cart.getcode(), bean2cart, n.ordine, bean2cart.getquantity(), bean2cart.getavailability()); - doRetrieveAllSearchComics(id); - doRetrieveAllSearchgadget(id); - doRetrieveByKey(id); - doUpdateC(bean, code); - doUpdateG(bean, code); - doDeleteComics(id); - doDeleteGadget(id); - doSaveC(bean); - doSaveG(bean);
Precondizione	context: GestioneProdotti :: doSaveC(bean); pre: bean.code non deve essere presente nel database context: GestioneProdotti :: doSaveG(bean); pre: bean.code non deve essere presente nel database context: GestioneProdotti :: doUpdateC(bean, code); pre: code deve essere presente nel database context:

Progetto: e-comix	Versione: 1.2
Documento: ODD	Data: 30/11/2017

	GestioneProdotti :: doUpdateG(bean, code); pre: code deve essere presente nel database context: GestioneProdotti :: doSaveC(bean); pre: bean.code non deve essere presente nel database context: GestioneProdotti :: doSaveG(bean); pre: bean.code non deve essere presente nel database
Post-Condizione	
Invariante	context: CartControl:: Shop(n.ordine, data, e-mail, prezzo); pre: prezzo > 0

GestioneOrdini

Nome	GestioneOrdini
Descrizione	Questo pacchetto fornisce diverse funzionalità che permettono di gestire gli ordini.
Signature dei Metodi	- doRetrieveAllAdmin(); - doRetrieveAllOrderComics(id); - doRetrieveAllOrderGadget(id); - doRetrieveAll(Email);
Precondizione	
Post-Condizione	
Invariante	

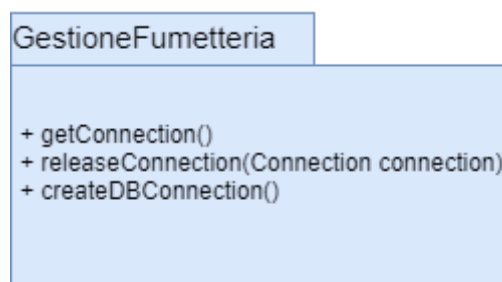
GestioneUtenti

Nome	Gestione Utenti
Descrizione	Questa class fornisce la funzione di Logout
Signature dei Metodi	+ getClient(Email); + doUpdate(Client, Email); + doSave(Client); + getCient(); + doDelete(Email);

Progetto: e-comix	Versione: 1.2
Documento: ODD	Data: 30/11/2017

	+ check(Email, Password). + mailus(email, object, message)
Precondizione	context: GestionUtenti :: doSave(Client); pre: Client.email non deve essere presente nel database
Post-Condizione	
Invariante	

2.5 PK - Storage



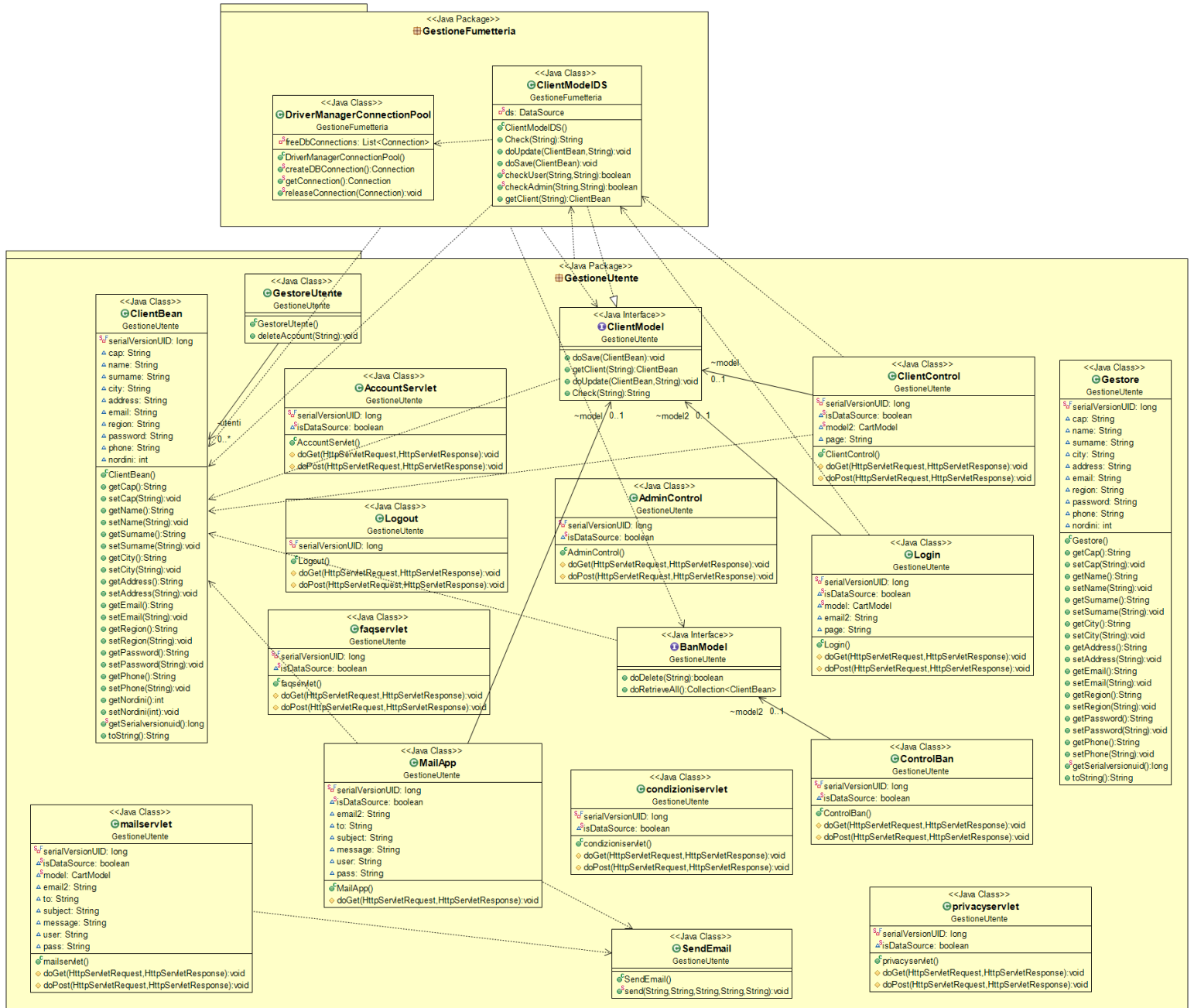
GestioneFumetteria

Nome	Gestione Utenti
Descrizione	Questo pacchetto rappresenta il gestore del pool di connessioni con il database.
Signature dei Metodi	+ getConnection() + releaseConnection(Connection connection) + createDBConnection()
Precondizione	
Post-Condizione	
Invariante	

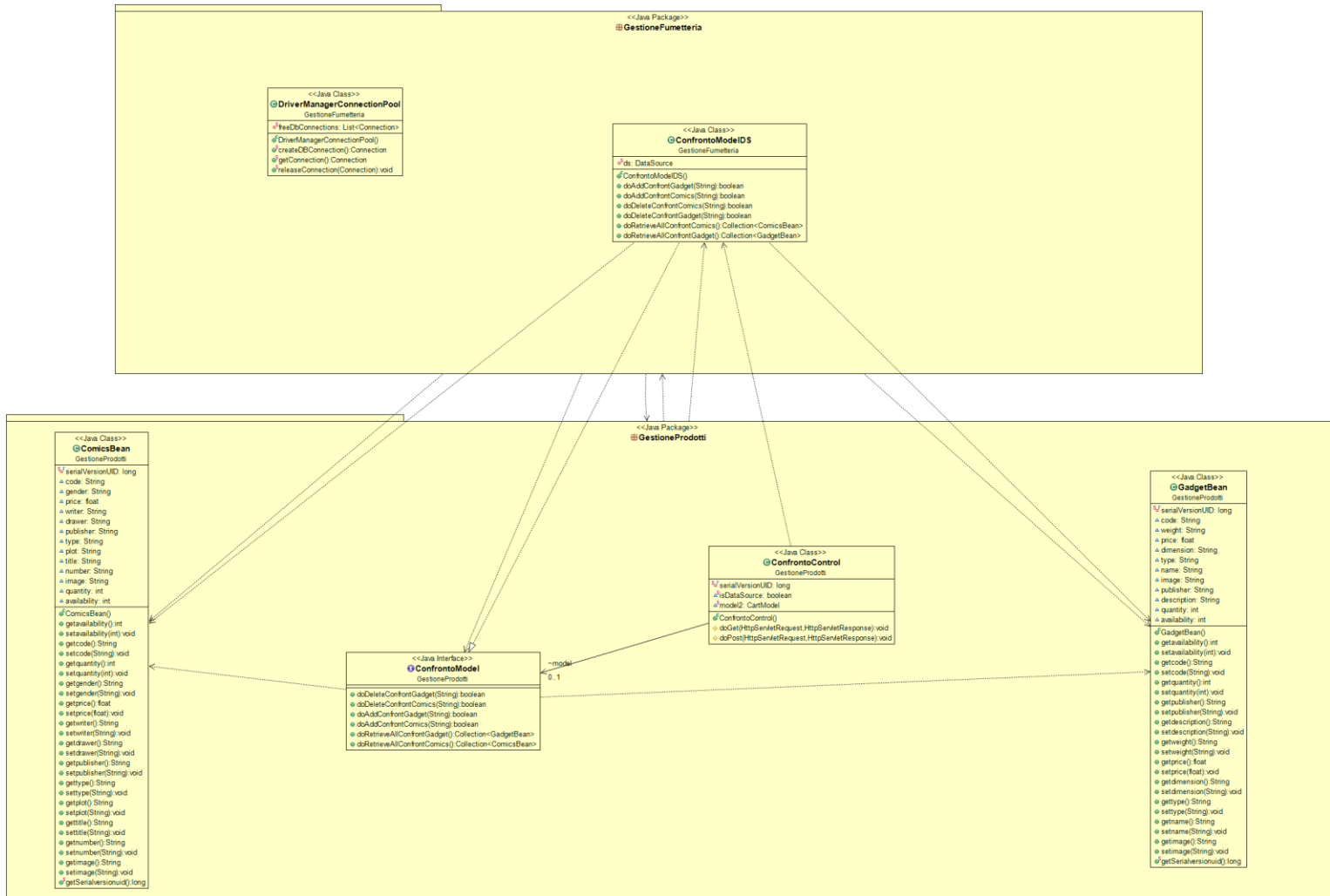
3. Interfacce delle classi

I seguenti diagrammi dei packages sono stati realizzati utilizzando ObjectAID, un plugin del framework di Eclipse, dopo aver completato lo sviluppo.

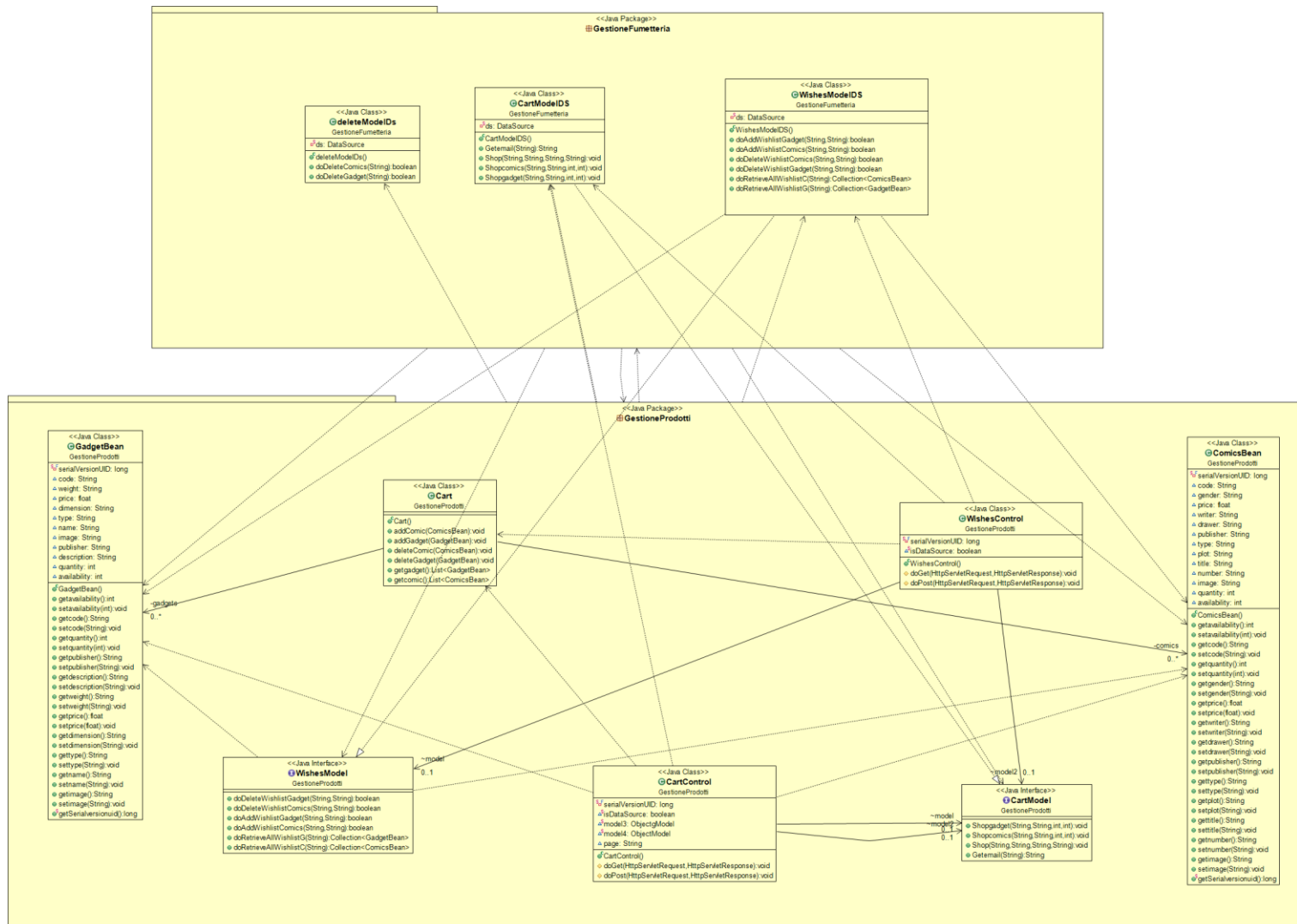
3.1 CD GestioneUtente



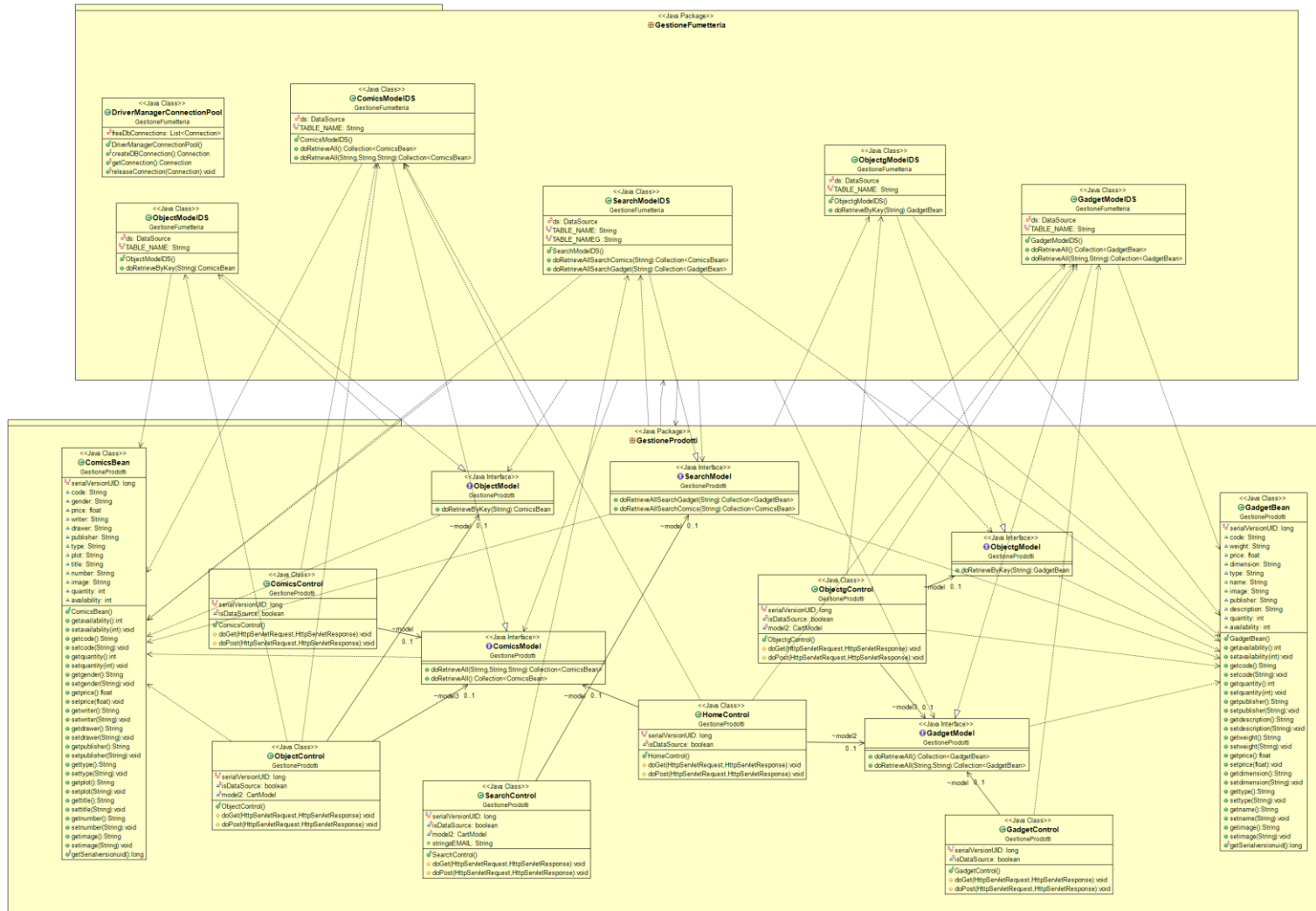
3.2.1 CD GestioneProdotti (Confronto)



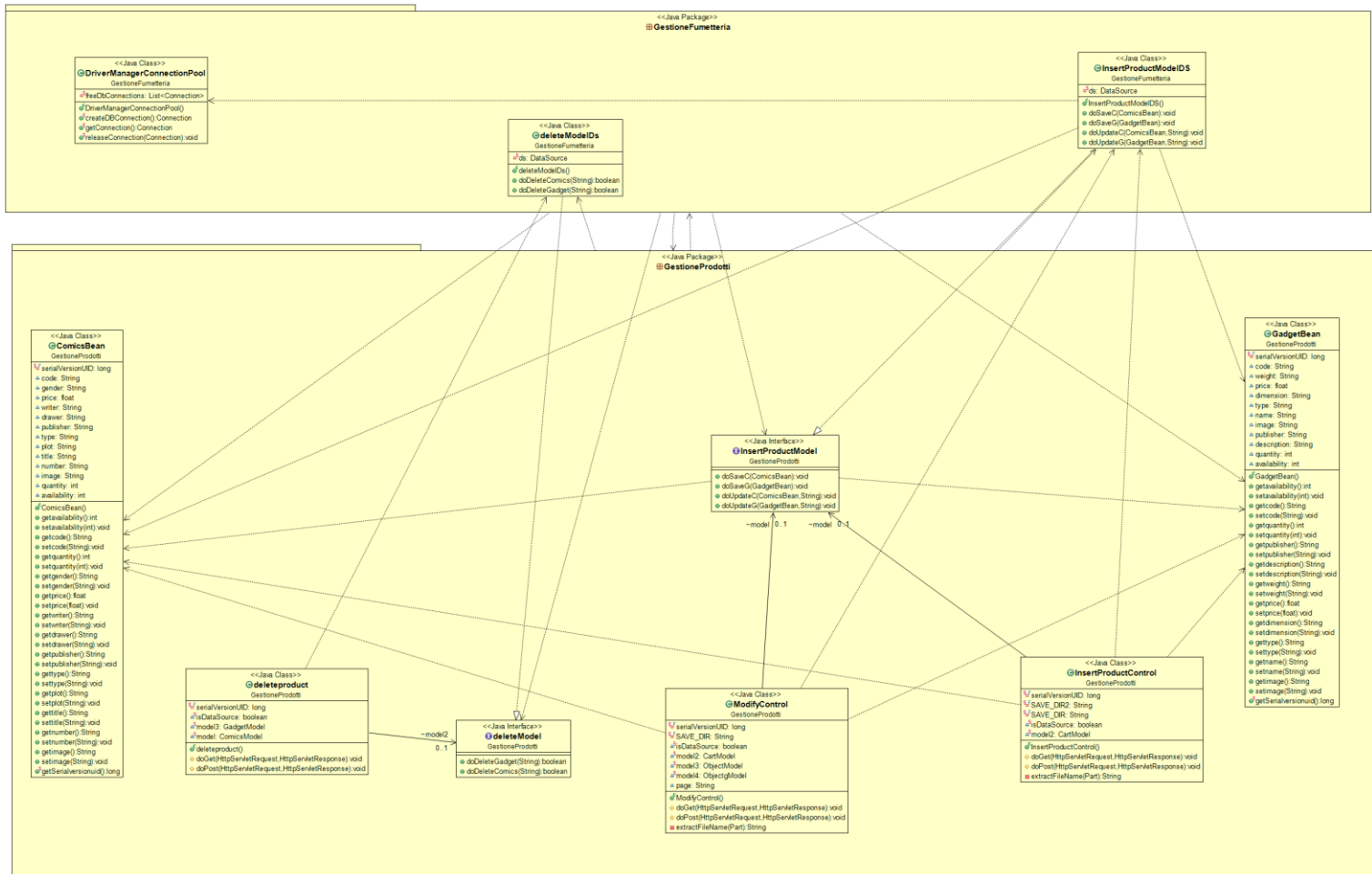
3.2.2 CD GestioneProdotti (Carrello + Wishlist)



3.2.3 CD GestioneProdotti (Visualizza + Ricerca)

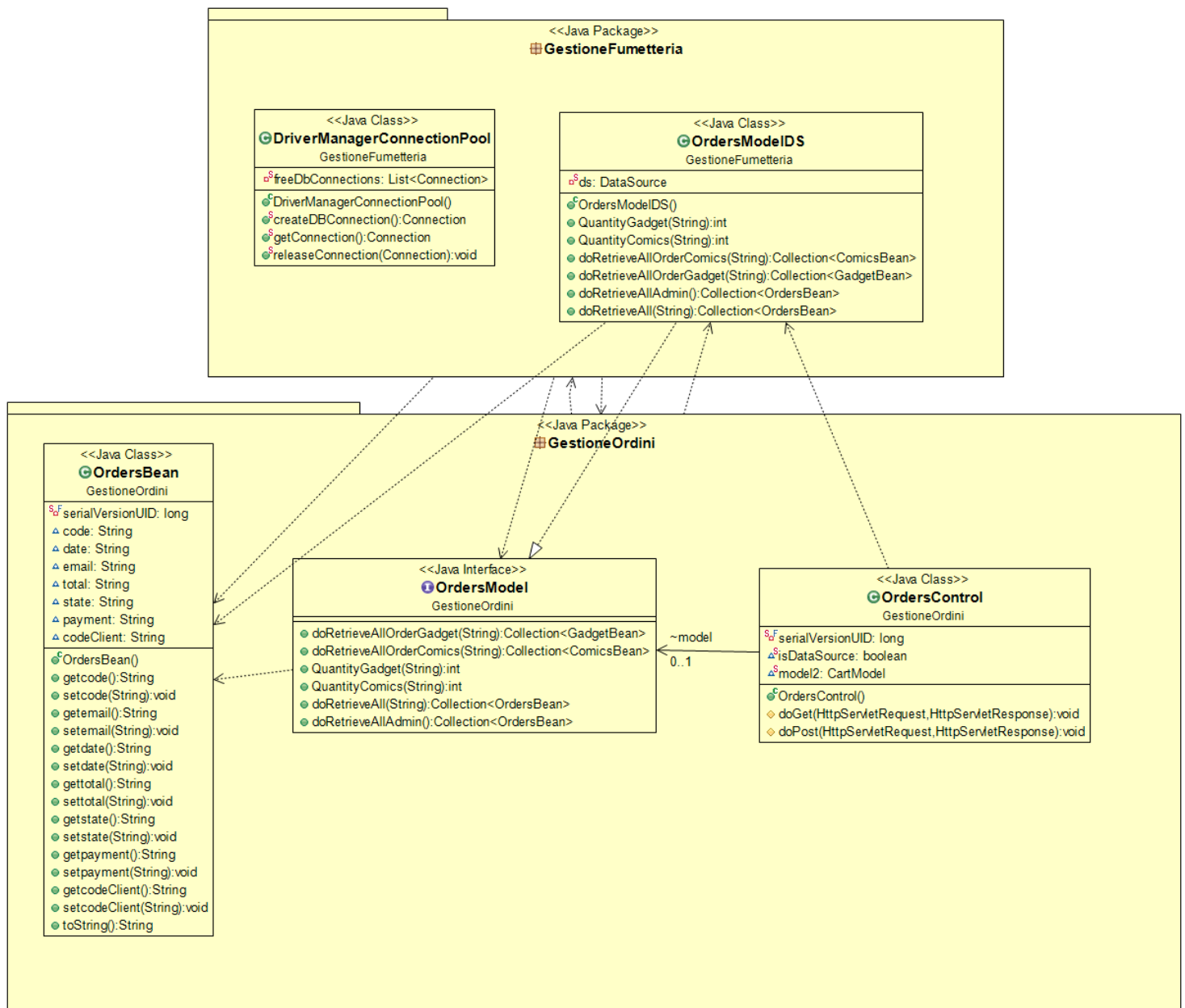


3.2.4 CD GestioneProdotti (Eliminazione/Modifica/Inserimento)



Progetto: e-comix	Versione: 1.2
Documento: ODD	Data: 30/11/2017

3.3 CD GestioneOrdini



Progetto: e-comix	Versione: 1.2
Documento: ODD	Data: 30/11/2017

4. Glossario

ObjectAID: Plug-in Eclipse.

MySQL: Software per la gestione di un database (Structured Query Language).

HTML: Linguaggio di mark-up per pagine web.

CSS: Linguaggio usato per definire la formattazione di pagine web.

Eclipse: ambiente di sviluppo integrato multi-linguaggio e multi-piattaforma.