Università degli Studi di Salerno
Corso di Ingegneria del Software

SocialNotes
Object Design Document
Versione 1.0

Data: 26/01/2022

Partecipanti:

Nome	Matricola
Alfonso Califano	0512109653
Armando Caso	0512106453
Francesco Di Lauro	0512106873
Simone Della Porta	0512109134

Revision History

Data	Versione	Descrizione	Autore
26/01/2022	1.0	Prima stesura del documento	Membri del team

Indice

1. INTRODUZIONE

1.1 Object Design Trade-Offs

1.1.1 Sicurezza vs Costi

A causa dei tempi di sviluppo ridotti e del budget per il progetto ridotto, si sceglie di limitare la sicurezza implementandola solamente attraverso la cifratura della password degli utenti della piattaforma. La sicurezza dei dati personali sarà preservata attraverso l'utilizzo di credenziali di accesso all'area personale(Username o Email, Password).

1.1.2 Tempo di risposta vs Hardware

Il tempo di risposta è un aspetto fondamentale per la piattaforma e per la sua utenza, esso verrà comunque influenzato anche dall'hardware sul quale verrà installato il sistema.

1.1.3 Persistenza vs Efficienza

Si è scelto di optare per una memorizzazione persistente dei dati mediante un DBMS per un database relazionale, aumentando il tempo di risposta e quindi diminuendo l'efficienza.

1.1.4 Comprensibilità vs Tempo

Il codice deve essere quanto più comprensibile possibile per favorore la manutenibilità e quindi eventuali modifiche e risoluzione di problemi futuri. Il codice quindi dovrà essere accompagnato da commenti che permettono di comprendere meglio le operazioni effettuate. Questa caratteristica richiederà più tempo per lo sviluppo del progetto.

1.2 Riferimenti

Si fa riferimento ai documenti:

- Requirements Analysis Document
- System Design Document

2. PACKAGES AND CLASS INTERFACES

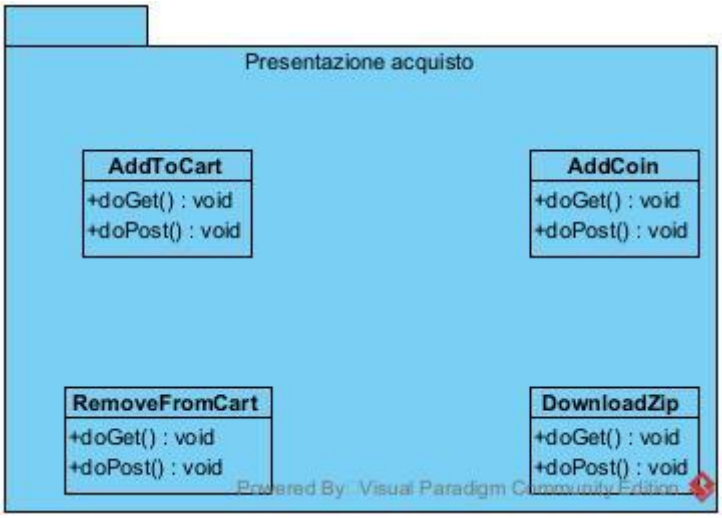
2.1 Package

Con la fase di System Decomposition sono stati individuati i package:

- Per il livello view – control :
    1. Presentazione acquisto
    2. Presentazione materiale
    3. Presentazione profilo
    4. Presentazione news
    5. Presentazione chat

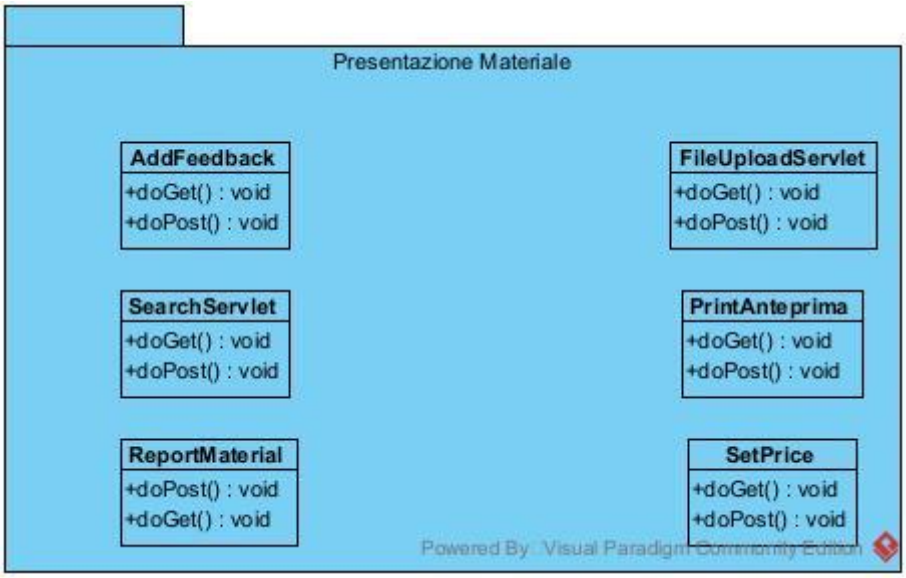
- Per il livello di logica di business:
    6. Chat
    7. News
    8. Profilo
    9. Materiale
    10. Acquisto

2.1.1 Presentazione acquisto



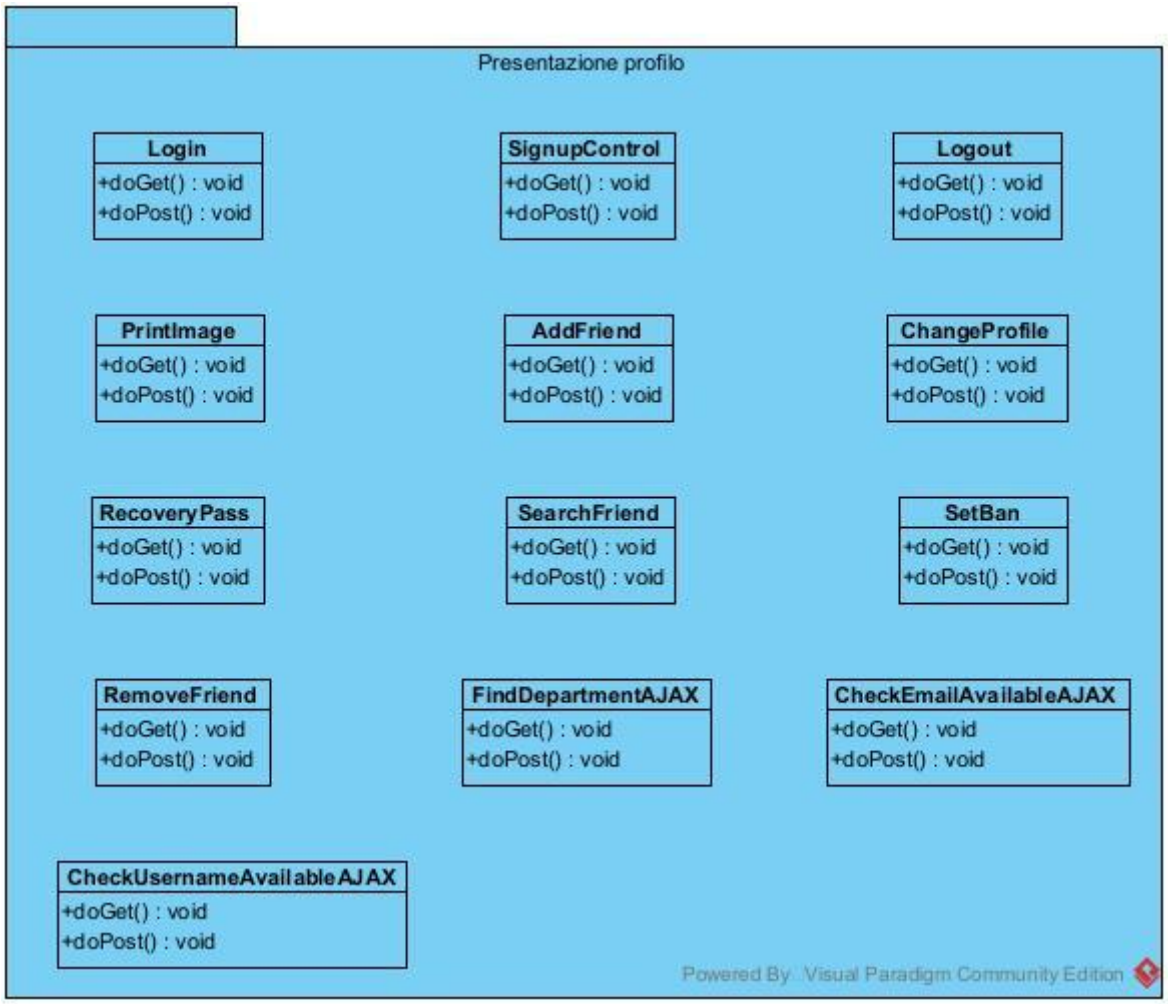
- **AddToCart:** è la servlet che permette di aggiungere un materiale al carrello
- **RemoveFromCart:** è la servlet che permette di eliminare un prodotto dal carrello
- **DownloadZip:** è la servlet che permette di effettuare il download del materiale acquistato
- **AddCoin:** è la servlet che offre le funzionalità per incrementare il numero di coin dell'utente quando esso effettua l'acquisto di questi ultimi

2.1.2 Presentazione materiale



- **AddFeedback:** è la servlet che permette di aggungere un feedback ad un materiale
- **FileUploadServlet:** è la servlet che permette di agguingere un nuovo materiale alla piattaforma
- **SearchServlet:** è la servlet che permette di effettuare la ricerca del materiale in base ad una determinata keyword
- **PrintAteprima:** è la servlet che permette di visualizzare l'anteprima di un materiale quando lo si visualizza
- **ReportMaterial:** è la servlet che permette al Notes Manager di segnalare un materiale e conseguentemente eliminarlo dalla piattaforma
- **SetPrice:** è la servlet che permette al Notes Manager di approvare un materiale rendendolo visibile agli utenti della piattaforma ed impostare un prezzo ad esso

2.1.3 Presentazione profilo

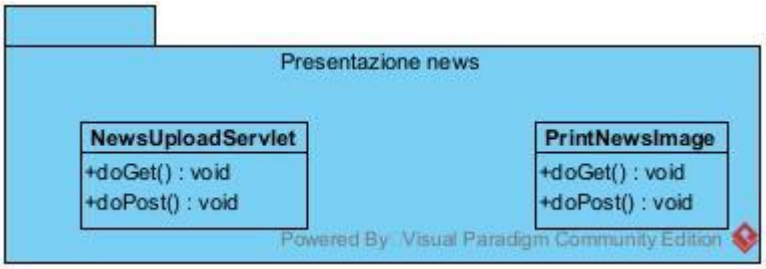


- **Login:** è la servlet che permette di efettuare le operazioni di autenticazione alla piattaforma
- **SignupControl:** è la servlet che permette di effettuare le operazioni di registrazione alla piattaforma
- **Logout:** è la servlet che permette di effettuare le operazioni di logout
- **PrintImage:** è la servlet che permette di visualizzare l'immagine di profilo di un utente
- **AddFriend:** è la servlet che permette di aggiungere un altro utente tra gli amici
- **ChangeProfile:** è la servlet che permette di effettuare la modifica dei dati personali
- **RecoveryPass:** è la servlet che permette di effettuare il recupero della password nel caso in cui non si ricordi la password
- **SearchFriend:** è la servlet che permette di effettuare la ricerca di utenti tramite una keyword
- **SetBan:** è la servlet che permette allo Users Manager di bannare un utente dal sito
- **RemoveFriend:** è la servlet che permette di rimuovere un utente dalla lista degli

amici

- **FindDepartmentAJAX:** è la servlet che permette di recuperare i dipartimenti di un università al momento della registrazione di un utente facendo utilizzo di AJAX
- **CheckEmailAvailableAJAX:** è la servlet che permette di verificare che una email non sia già utilizzata al momento della registrazione di un utente tramite l'utilizzo di AJAX
- **CheckUsernameAvailableAJAX:** è la servlet che permette di verificare che uno username non sia già utilizzato al momento della registrazione tramite l'utilizzo di AJAX

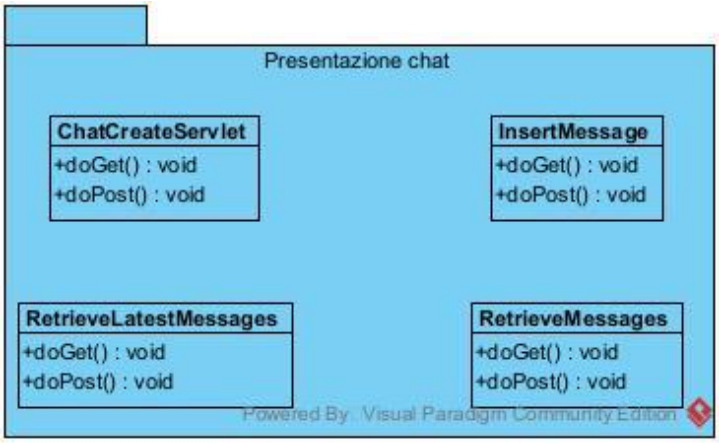
2.1.4 Presentazione News



- **NewsUploadServlet:** è la servlet che permette al News Manager di inserire nuove news all'interno della piattaforma.

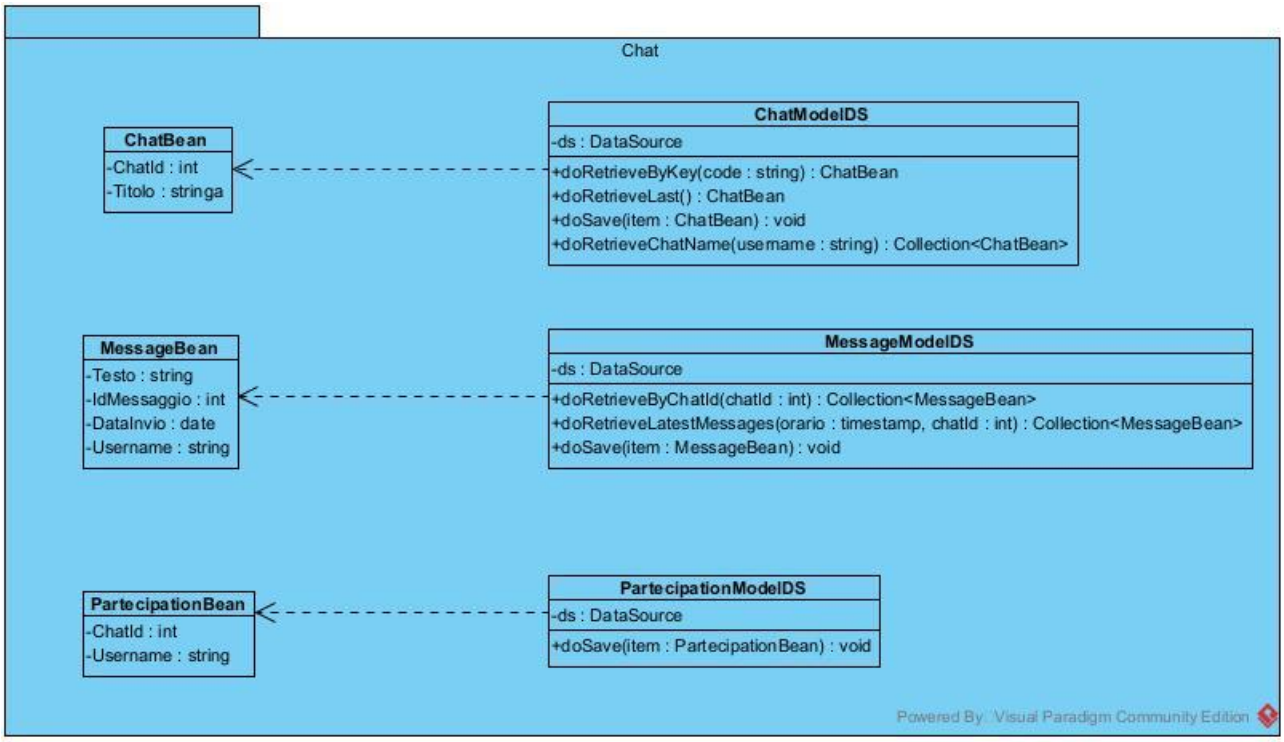
- **PrintNewsImage:** è la servlet che permette di visualizzare l'immagine associati ad una news.

2.1.5 Presentazione Chat



- **ChatCreateServlet:** è la servlet che permette ad un utente di creare una chat per comunicare con altri utenti.
- **InsertMessage:** è la servlet che permette ad un utente di inserire un messaggio di testo all'interno di una chat esistente.
- **RetrieveLatestMessages:** è la servlet che permette di ottenere l'ultimo messaggio inviato all'interno della chat.
- **RetrieveMessages:** è la servlet che permette di ottenere tutti i messaggi che sono stati inviati all'interno di una chat.

2.1.6 Chat Class Interfaces



- **ChatBean:** rappresenta l'entità Chat all'interno del database.
- **MessageBean:** rappresenta l'entità Messaggio all'interno del database.
- **PartecipationBean:** rappresenta l'entità Partecipazione all'interno del database.

Nome Classe	ChatModelDs
Descrizione	Questa classe è un manager che si occupa di interagire con il database e permette di interrogare la tabella Chat

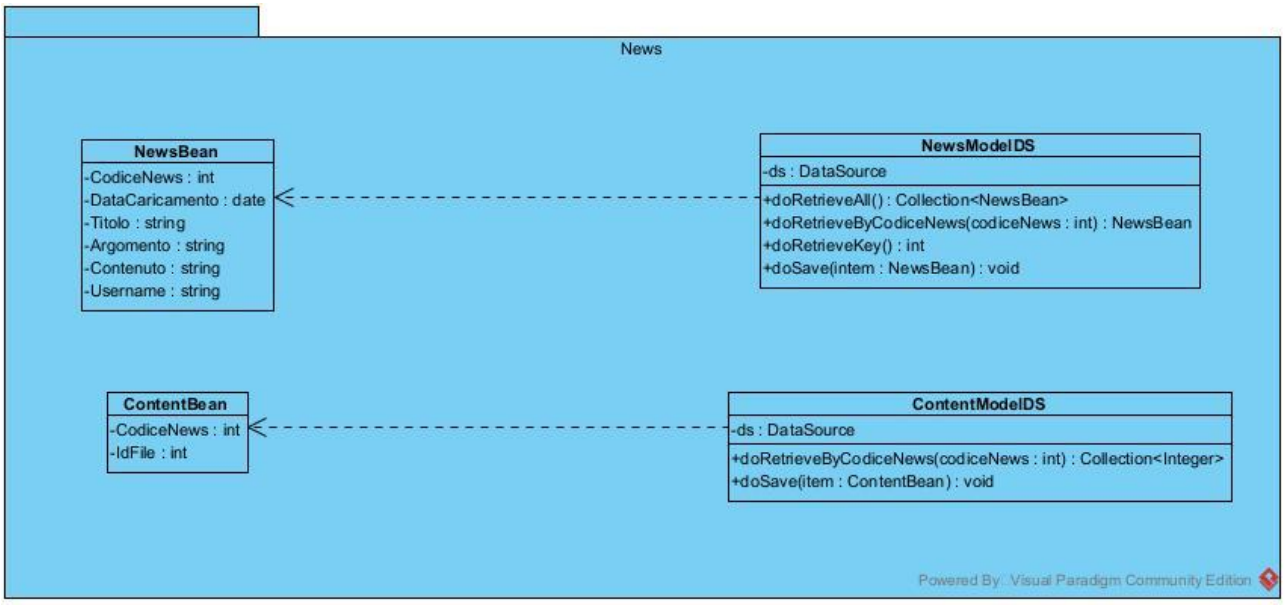
Pre-Condizione	**Context** ChatModelDS **::** doRetrieveByKey(String code) **Pre:** code!=null and code!="" **Context** ChatModelDS **::** doRetrieveLast() **Pre:** nessuna pre-condizione **Context** ChatModelDS **::** doSave(ChatBean item) **Pre:** item != null **Context** ChatModelDS **::** doRetrieveChatName(String username) **Pre:** username != null and username != ""
Post-Condizione	**Context** ChatModelDS **::** doRetrieveByKey(String code) **Post:** return ChatBean or null se non esiste **Context** ChatModelDS **::** doRetrieveLast() **Post:** return ChatBean or null se non esiste alcuna istanza della tabella Chat nel database **Context** ChatModelDS **::** doSave(ChatBean item) **Post:** viene inserita una nuova istanza nella tabella Chat del database con i valori contenuti in item **Context** ChatModelDS **::** doRetrieveChatName(String username) **Post:** return Collection<ChatBean>
Invarianti	

Nome Classe	**MessageModelDS**
Descrizione	Questa classe è un manager che si occupa di interagire con il database e permette di interrogare la tabella Messaggio
Pre-Condizione	**Context** MessageModelDS **::** doRetrieveByChatId(int chatid) **Pre:** chatid >= 0 **Context** MessageModelDS **::** doRetrieveLatestMessages (Timestamp orario, int chatid) **Pre:** orario!=null and chatid >= 0 **Context** MessageModelDS **::** doSave(MessageBean item) **Pre:** item != null
Post-Condizione	**Context** MessageModelDS **::** doRetrieveByChatId(int chatid) **Post:** return Collection<MessageBean> **Context** MessageModelDS **::** doRetrieveLatestMessages (Timestamp orario, int chatid) **Post:** return Collection<MessageBean> **Context** MessageModelDS **::** doSave(MessageBean item)

	Post: viene inserita una nuova istanza nella tabella Messaggio del database con i valori contenuti in item
Invarianti	

Nome Classe	**PartecipationModelDS**
Descrizione	Questa classe è un manager che si occupa di interagire con il database e permette di interrogare la tabella Partecipazione
Pre-Condizione	**Context** PartecipationModelDS **::** doSave(PartecipationBean item) **Pre:** item != null
Post-Condizione	**Context** PartecipationModelDS **::** doSave(PartecipationBean item) **Post:** viene inserita una nuova istanza nella tabella Partecipazione del database con i valori contenuti in item
Invarianti	

2.1.7 News Class Interfaces

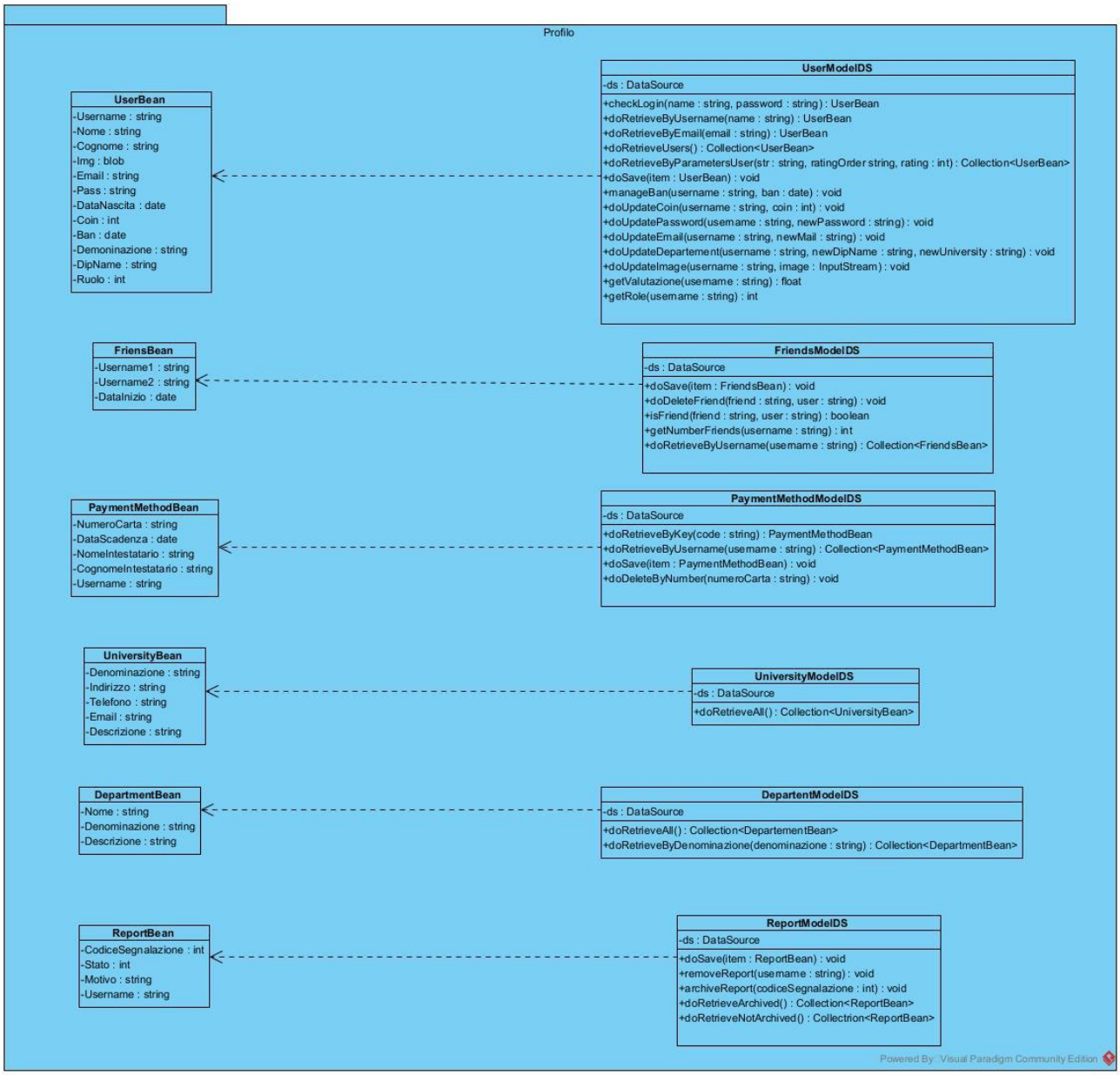


- **NewsBean:** rappresenta l'entità News all'interno del database.
- **ContentBean:** rappresenta l'entità Content all'interno del database.

Nome Classe	NewsModelDS
Descrizione	Questa classe è un manager che si occupa di interagire con il database e permette di interrogare la tabella News
Pre-Condizione	**Context** NewsModelDS **::** doRetrieveAll() **Pre:** nessuna pre-condizione **Context** NewsModelDS **::** doRetrieveByCodiceNews(int codiceNews) **Pre:** codiceNews >= 0 **Context** NewsModelDS **::** doRetrieveKey() **Pre:** nessuna pre-condizione **Context** NewsModelDS **::** doSave(NewsBean item) **Pre:** item != null
Post-Condizione	**Context** NewsModelDS **::** doRetrieveAll() **Post:** return Collection<NewsBean> **Context** NewsModelDS **::** doRetrieveByCodiceNews(int codiceNews) **Post:** return NewsBean or null se non esiste **Context** NewsModelDS **::** doRetrieveKey() **Post:** return int che corrisponde al codice dell'ultima news inserita **Context** NewsModelDS **::** doSave(NewsBean item) **Post:** viene inserita una nuova istanza nella tabella News del database con i valori contenuti in item
Invarianti	

Nome Classe	ContentModelDS
Descrizione	Questa classe è un manager che si occupa di interagire con il database e permette di interrogare la tabella Contenuto
Pre-Condizione	**Context** ContentModelDS **::** doRetrieveByCodiceNews(int codiceNews) **Pre:** codiceNews >= 0 **Context** ContentModelDS **::** doSave(ContentBean item) **Pre:** item != null
Post-Condizione	**Context** ContentModelDS **::** doRetrieveByCodiceNews(int codiceNews) **Post:** return Collection<Integer>, tale collezione contiene gli ID dei file contenuti nella News **Context** ContModelDS **::** doSave(ContentBean item) **Post:** viene inserita una nuova istanza nella tabella Contenuto del database con i valori contenuti in item
Invarianti	

2.1.8 Profilo Class Interfaces



- **UserBean:** rappresenta l'entità Utente all'interno del database.
- **FriendsBean:** rappresenta l'entità Amicizia all'interno del database.
- **PaymentMethodBean:** rappresenta l'entità MetodoPagamento del database.
- **UniveristyBean:** rappresenta l'entità Università all'interno del database.
- **DepartmentBean:** rappresenta l'entità Dipartimento all'interno del database.
- **ReportBean:** rappresenta l'entità Segnalazione all'interno del database.

Nome Classe	UserModelDS
Descrizione	Questa classe è un manager che si occupa di interagire con il database e permette di interrogare la tabella Utente
Pre-Condizione	**Context** UserModelDS **::** checkLogin(String name,String password) **Pre:** name!=null and name!=”” and password!=null and password!=”” **Context** UserModelDS **::** doRetrieveByUsername(String name) **Pre:** name!=null and name!=”” **Context** UserModelDS **::** doRetrieveByEmail(String email) **Pre:** email!=null and email!=”” **Context** UserModelDS :: doRetrieveUsers() **Pre:** nessuna pre-condizione **Context** UserModelDS :: doRetrieveByParametersUser(String str,String ratingOrder,int rating) **Pre:** str!=null and (ratingOrder==”ASC” or ratingOrder==”DESC” or ratingOrder==”novalue”) and rating >=0 and rating<=5 **Context** UserModelDS :: doSave(UserBean item) **Pre:** item!=null **Context** UserModelDS :: manageBan(String username,Date ban) **Pre:** username!=null and username!=”” and ban!=null and ban>data attuale **Context** UserModelDS :: doUpdateCoin(String username,int coin) **Pre:** username!=null and username!=”” and coin>0 **Context** UserModelDS :: doUpdatePassword(String username,String newPassword) **Pre:** username!=null and username!=”” and newPassword!=null and newPassword!=”” **Context** UserModelDS :: doUpdateEmail(String username,String newMail) **Pre:** username!=null and username!=”” and newMail!=null and newMail!=”” **Context** UserModelDS :: doUpdateDepartment(String username,String newDipName,String newUniversity) **Pre:** username!=null and username!=”” and newDipName!=null and newDipName!=”” and newUniversity!=null and newUniversity!=”” **Context** UserModelDS :: doUpdateImage(String username,InputStream image) **Pre:** username!=null and username!=”” and image!=null

	Context UserModelDS :: getValutazione(String username) **Pre:** username!=null and username!=”” **Context** UserModelDS :: getRole(String username) **Pre:** username!=null and username!=””
Post-Condizione	**Context** UserModelDS **::** checkLogin(String name,String password) **Post:** return UserBean or null se non esiste,l’oggetto UserBean contiene tutte le informazioni sull’utente che ha effettuato il login **Context** UserModelDS **::** doRetrieveByUsername(String name) **Post:** return UserBean or null se non esiste,l’oggetto UserBean contiene tutte le informazioni sull’utente che ha come Username name **Context** UserModelDS **::** doRetrieveByEmail(String email) **Post:** return UserBean or null se non esiste,l’oggetto UserBean contiene tutte le informazioni sull’utente che ha come Email email **Context** UserModelDS :: doRetrieveUsers() **Post:** return Collection<UserBean>,tale collection contiene tutti gli utenti (studenti) della piattaforma **Context** UserModelDS :: doRetrieveByParametersUser(String str,String ratingOrder,int rating) **Post:** return Collection<UserBean>,tale collezione contiene gli utenti che rispettano i criteri utilizzati per effettuare la ricerca **Context** UserModelDS :: doSave(UserBean item) **Post:** viene inserita una nuova istanza nella tabella Utente con le informazioni contenute in item **Context** UserModelDS :: manageBan(String username,Date ban) **Post:** viene aggiornato il valore dell’attributo Ban a ban dell’istanza della tabella Utente che ha come Username username **Context** UserModelDS :: doUpdateCoin(String username,int coin) **Post:** viene aggiornato il valore dell’attributo Coin a coin dell’istanza della tabella Utente che ha come Username username **Context** UserModelDS :: doUpdatePassword(String username,String newPassword) **Post:** viene aggiornato il valore dell’attributo Pass a newPassword dell’istanza della tabella Utente che ha come Username username **Context** UserModelDS :: doUpdateEmail(String username,String newMail) **Post:** viene aggiornato il valore dell’attributo Email a newMail dell’istanza della tabella Utente che ha come Username username

	Context UserModelDS :: doUpdateDepartment(String username,String newDipName,String newUniversity) **Post:** viene aggiornato il valore degli attributi dipName e Denominazione a newDipName e newUniversity dell'istanza della tabella Utente che ha come Username username **Context** UserModelDS :: doUpdateImage(String username,InputStream image) **Post:** viene aggiornato il valore dell'attributo Img a image dell'istanza della tabella Utente che ha come Username username **Context** UserModelDS :: getValutazione(String username) **Post:** return int,il valore restituito rappresenta la valutazione media del materiale caricato dall'utente che ha come Username username **Context** UserModelDS :: getRole(String username) **Post:** return int,il valore restituito rappresenta il ruolo dell'utente
Invarianti	

Nome Classe	**FriendsModelDS**
Descrizione	Questa classe è un manager che si occupa di interagire con il database e permette di interrogare la tabella Amicizia
Pre-Condizione	**Context** FriendsModelDS **::** doSave(Friendsbean item) **Pre:** item!=null **Context** FriendsModelDS **::** doDeleteFriend(String friend, String user) **Pre:** friend!=null or friend!="" AND user!=null or user!="" **Context** FriendsModelDS **::** isFriend(String friend, String user) **Pre:** friend!=null or friend!="" AND user!=null or user!="" **Context** FriendsModelDS **::** getNumberFriends(String username) **Pre:** username!=null AND username!="" **Context** FriendsModelDS **::** doRetrieveByUsername(String username) **Pre:** username!=null AND username!=null
Post-Condizione	**Context** FriendsModelDS **::** doSave(Friendsbean item) **Post:** viene inserita una nuova istanza nella tabella Amicizia del database con i valori contenuti in item **Context** FriendsModelDS **::** doDeleteFriend(String friend, String user) **Post:** viene rimossa una tupla dalla tabella Amicizia dove il campo username1 = *friend* e il campo username2 = *user* **Context** FriendsModelDS **::** isFriend(String friend, String user) **Post:** return boolean, tale valore è true se *friend* e *user* sono amici, altrimenti il valore è false.

	Context FriendsModelDS **::** getNumberFriends(String username) **Post:** return int , tale valore rappresenta il numero di amici dell'utente il cui valore nel campo username è *username* **Context** FriendsModelDS **::** doRetrieveByUsername(String username) **Post:** return Collection<FriendsBean> , la collezione restituta è una collezione contente tutti gli amici dell'utente che ha nel campo username il valore *username*
Invarianti	

Nome Classe	**PaymentMethodModelDS**
Descrizione	Questa classe è un manager che si occupa di interagire con il database e permette di interrogare la tabella MetodoPagamento
Pre-Condizione	**Context** PaymentMethodModelDS **::** doRetrievebyKey(String code) **Pre:** code.length() = 16 AND code!=null AND code !="" **Context** PaymentMethodModelDS **::** doRetrieveByUsername(String username) **Pre:** username!=null AND username!="" **Context** PaymentMethodModelDS **::** doSave (PaymentMethodBean item) **Pre:** item!=null **Context** PaymentMethodModelDS **::** doDelete(String numeroCarta) **Pre:** numeroCarta.length() = 16 AND numeroCarta!=null AND numeroCarta != ""
Post-Condizione	**Context** PaymentMethodModelDS **::** doRetrievebyKey(String code) **Post:** return PaymentMethodBean or null se non esiste **Context** PaymentMethodModelDS **::** doRetrieveByUsername(String username) **Post:** return Collection<PaymenntMethodBean> , dove tale Collection è una collezione di tutti i metodi di pagamento associati ad un Utente che ha Username *username* **Context** PaymentMethodModelDS **::** doSave (PaymentMethodBean item) **Post:** viene inserita una nuova istanza nella tabella MetodoPagamento del database con i valori contenuti in item **Context** PaymentMethodModelDS **::** doDelete(String numeroCarta) **Post:** viene rimossa una tupla dalla tabella MetodoPagamento dove il campo NumeroCarta = *NumeroCarta*
Invarianti	

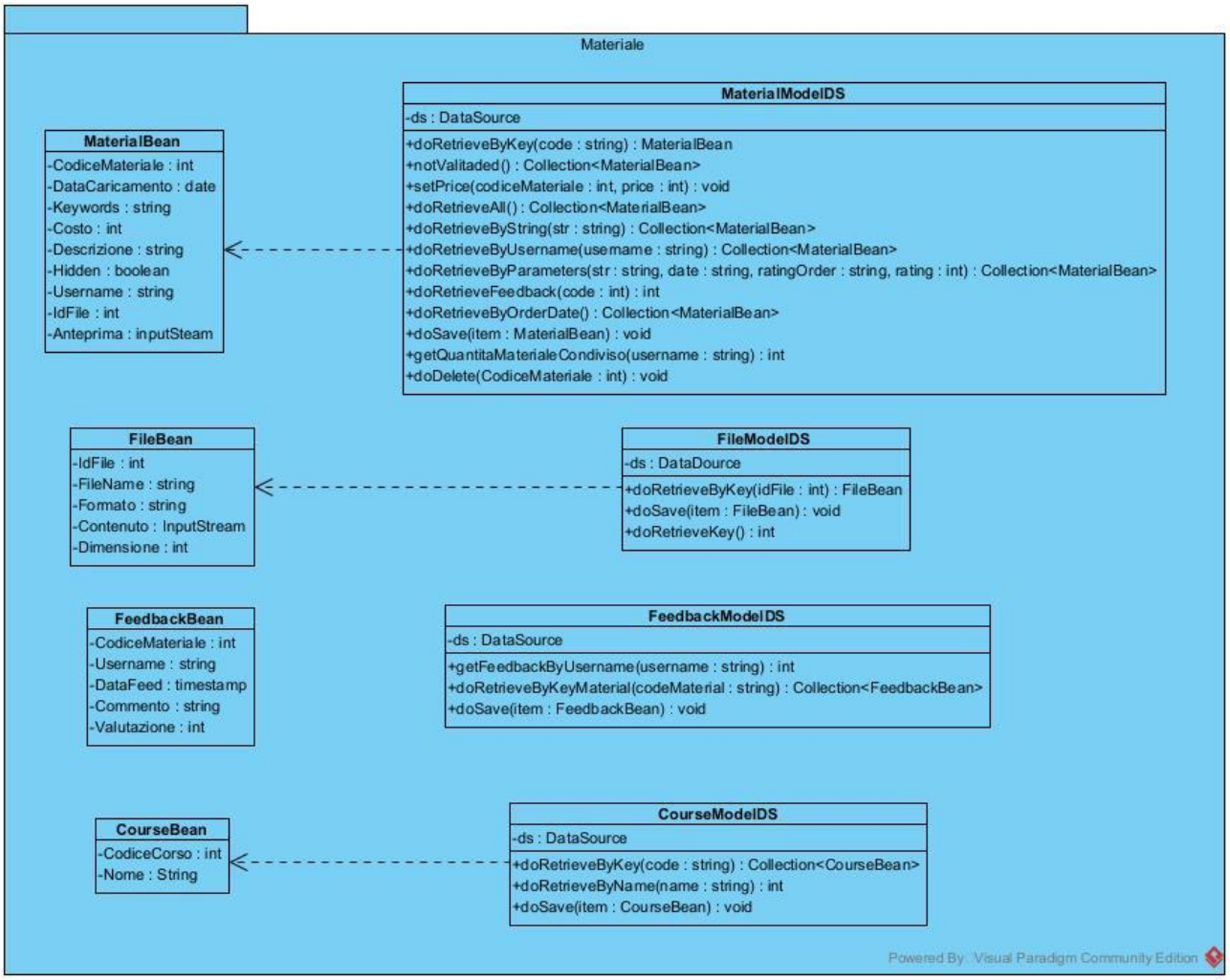
Nome Classe	UniversityModelDS
Descrizione	Questa classe è un manager che si occupa di interagire con il database e permette di interrogare la tabella Universita
Pre-Condizione	**Context** UniversityModelDS **::** doRetrieveAll() **Pre:** nessuna pre-condizione
Post-Condizione	**Context** UniversityModelDS **::** doRetrieveAll() **Post:** return Collection<UniversityBean>
Invarianti	

Nome Classe	DepartmentModelDS
Descrizione	Questa classe è un manager che si occupa di interagire con il database e permette di interrogare la tabella Dipartimento
Pre-Condizione	**Context** DepartmentModelDS **::** doRetrieveAll() **Pre:** nessuna pre-condizione **Context** DepartmentModelDS **::** doRetrieveByDenominazione(String denominazione) **Pre:** denominazione!=null and denominazione!=””
Post-Condizione	**Context** DepartmentModelDS **::** doRetrieveAll() **Post:** return Collection<DepartmentBean> **Context** DepartmentModelDS **::** doRetrieveByDenominazione(String denominazione) **Post:** return Collection<DepartmentBean>,la collezione contiene tutte le istanze della tabella Dipartimento che hanno come Denominazione denominazione,cioè contiene tutti i dipartimenti associati ad un università
Invarianti	

Nome Classe	ReportModelDS
Descrizione	Questa classe è un manager che si occupa di interagire con il database e permette di interrogare la tabella Segnalazione
Pre-Condizione	**Context** ReportModelDS **::** doSave (ReportBean item) **Pre:** item!=null **Context** ReportModelDS **::** removeReport(String username) **Pre:** username!=null and username!=”” **Context** ReportModelDS **::** archiveReport(int codiceSegnalazione) **Pre:** codiceSegnalazione >= 0

	Context ReportModellDS :: doRetrieveNotArchived() **Pre:** nessuna pre-condizione **Context** ReportModelDS :: doRetrieveArchived() **Pre:** nessuna pre-condizione
Post-Condizione	**Context** ReportModelDS **::** doSave (ReportBean item) **Post:** viene inserita una nuova istanza nella tabella Segnalazione del database con i valori contenuti in item **Context** ReportModelDS **::** removeReport(int codiceSegnalazione) **Post:** elimina la tupla all'interno della tabella Segnalazione presente nel database dove il campo codiceSegnalazione = *codiceSegnalazione* **Context** ReportModelDS **::** archiveReport(int codiceSegnalazione) **Post:** imposta il valore 1(archiviato) al campo Stato della tupla presente nella tabella Segnalazione nel database dove codiceSegnalazione = *codiceSegnalazione* **Context** ReportModellDS :: doRetrieveNotArchived() **Post:** return Collection<ReportBean>,questa collezione contiene tutte le segnalazioni che sono nello stato "preso in considerazione" **Context** ReportModelDS :: doRetrieveArchived() **Post:** return Collection<ReportBean>,questa collezione contiene tutte le segnalazioni che sono nello stato "archiviato"
Invarianti	

2.1.9 Materiale Class Interfaces



- **MaterialBean**: rappresenta l'entità materiale all'interno del database
- **FileBean**: rappresenta l'entità file all'interno del database
- **FeedbackBean**: rappresenta l'entità Feedback all'interno del database
- **CourseBean**: rappresenta l'entità Corso all'interno del database

Nome Classe	MaterialModelDS
Descrizione	Questa classe è un manager che si occupa di interagire con il database e permette di interrogare la tabella Materiale
Pre-Condizione	**Context** MaterialModelDS **::** doRetrieveByKey(String code) **Pre:** code!=null and code!="" **Context** MaterialModelDS **::** notValidated() **Pre:** nessuna pre-condizione **Context** MaterialModelDS **::** setPrice(int codiceMateriale, int price) **Pre:** codiceMateriale>=0 AND price >=0 **Context** MaterialModelDS **::** doRetrieveByString(String str)

	Pre: str!=null AND str!=”” **Context** MaterialModelDS **::** doRetrieveByUsername(String username) **Pre:** username!=null AND username!=”” **Context** MaterialModelDS **::** doRetrieveByParameters(String str, String date, String ratingOrder, int rating) **Pre: (**str!=null and str!=””) AND (date=”ASC” OR date=”DESC” OR date=”novalue”) AND (ratingOrder=”ASC” OR ratingOrder=”DESC” OR ratingOrder=”novalue”) AND(rating>=0 and rating <=5) **Context** MaterialModelDS **::** doRetrieveFeedback (int code) **Pre:** code>=0 **Context** MaterialModelDS **::** doRetrieveByOrderDate () **Pre:** nessuna pre-condizione **Context** MaterialModelDS **::** doSave (MaterialBean item) **Pre:** item!=null **Context** MaterialModelDS **::** getQuantitaMaterialeCondiviso(String username) **Pre:** username!=null **Context** MaterialModelDS **::** doDelete(int codiceMateriale) **Pre:** codiceMateriale>=0
Post-Condizione	**Context** MaterialModelDS **::** doRetrieveByKey(String code) **Post:** return MaterialBean **Context** MaterialModelDS **::** notValidated () **Post:** return Collection<MaterialBean >, una collezione di MaterialBean che ha il campo hidden settato a true, ovvero materiale che non è stato ancora verificato **Context** MaterialModelDS **::** setPrice (int codiceMateriale, int price) **Post:** viene modificata l’istanza nella tabella Materiale corrispondente al codiceMateriale, aggiornando il suo prezzo a *price* e settando il suo campo hidden a false, indicando che il materiale è stato verificato **Context** MaterialModelDS **::** doRetrieveByString (String str) **Post:** return Collection<MaterialBean>, una collezione di MaterialBean, i cui campi Descrizione hanno corrispondenza con la stringa *str* oppure il corso del materiale ha corrispondenza con la stringa *str*. **Context** MaterialModelDS **::** doRetrieveByUsername (String username) **Post:** return Collection<MaterialBean>, una collezione di MaterialBean, il cui proprietario è l’utente con nome utente *username* e il cui campo hidden è

	settato a false, ovvero il materiale è stato verificato **Context** MaterialModelDS **::** doRetrieveByParameters(String str, String date, String ratingOrder, int rating) **Post:** return Collection<MaterialBean> di MaterialBean che contengono la stringa *str* come descrizione e vengono ordinati in base al *rating* e ai parametri date e ratingOrder che specificano se l'ordinamento SQL deve essere crescente o decrescente **Context** MaterialModelDS **::** doRetrieveFeedback (int code) **Post:** return la valutazione media del materiale nella tabella Materiale che ha codiceMateriale *code* oppure 0 se il Materiale non ha ancora alcun feedback **Context** MaterialModelDS **::** doRetrieveByOrderDate () **Post:** return Collection<MaterialBean>, una collezione di MaterialBean, il cui campo hidden è settato a false e seguendo un ordinamento SQL per il campo dataCaricamento di tipo discendente **Context** MaterialModelDS **::** doSave (MaterialBean item) **Post:** viene inserita una nuova istanza nella tabella News del database con i valori contenuti in item **Context** MaterialModelDS **::** getQuantitaMaterialeCondiviso(String username) **Post:** return la quantità del materiale condiviso dall'utente, se l'utente non ha condiviso materiale return 0 **Context** MaterialModelDS **::** doDelete(int codiceMateriale) **Post:** viene rimossa la tupla all'interno della tabella Materiale presente nel database, in cui il campo codiceMateriale = *codiceMateriale*
Invarianti	

Nome Classe	**FileModelDS**
Descrizione	Questa classe è un manager che si occupa di interagire con il database e permette di interrogare la tabella File
Pre-Condizione	**Context** FileModelDS **::** doRetrieveByKey(int idFile) **Pre:** idFile>=0 **Context** FileModelDS **::** doSave (FileBean item) **Pre:** item!=null **Context** FileModelDS :: doRetrieveKey() **Pre:** nessuna pre-condizione
Post-Condizione	**Context** FileModelDS **::** doRetrieveByKey(int idFile) **Post:** return FileBean

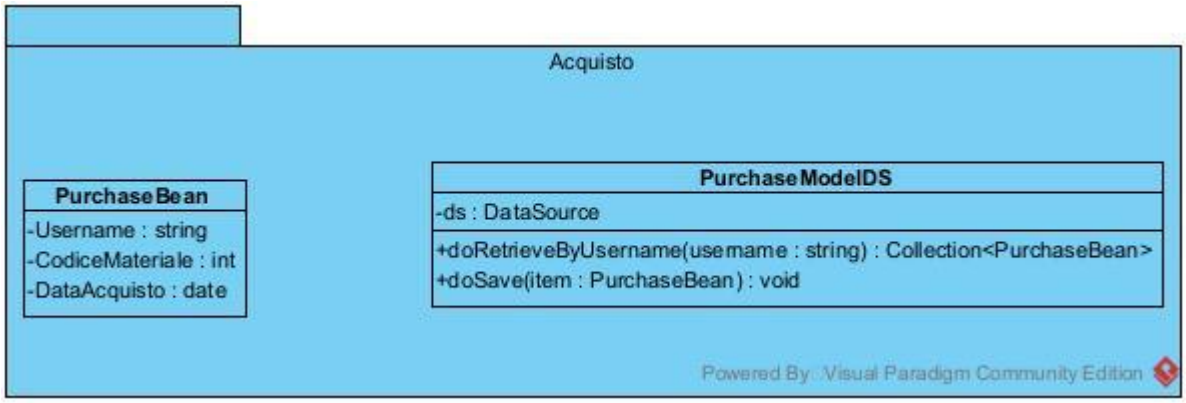
	Context FileModelDS **::** doSave (FileBean item) **Post:** viene inserita una nuova istanza nella tabella File del database con i valori contenuti in item **Context** FileModelDS :: doRetrieveKey() **Post:** return int che rappresenta la chiave primaria dell'ultimo file inserito
Invarianti	

Nome Classe	**FeedbackModelDS**
Descrizione	Questa classe è un manager che si occupa di interagire con il database e permette di interrogare la tabella Feedback
Pre-Condizione	**Context** FeedbackModelDS **::** doRetrieveByKeyMaterial(int codeMaterial) **Pre:** codeMaterial >= 0 **Context** FeedbackModelDS **::** doSave(FeedbackBean item) **Pre:** item != null **Context** FeedbackModelDS **::** getFeedbackByUsername(String username) **Pre:** username!=null and username!=""
Post-Condizione	**Context** FeedbackModelDS **::** doRetrieveByKeyMaterial(int codeMaterial) **Post:** return Collection <FeedbackBean> **Context** FeedbackModelDS **::** doSave(FeedbackBean item) **Post:** viene inserita una nuova istanza nella tabella Feedback del database con i valori contenuti in item **Context** FeedbackModelDS **::** getFeedbackByUsername(String username) **Post:** return int; l'intero restituito rappresenta il feedback relativo all'utente che ha come username la stringa inserita
Invarianti	

Nome Classe	**CourseModelDS**
Descrizione	Questa classe è un manager che si occupa di interagire con il database e permette di interrogare la tabella Corso
Pre-Condizione	**Context** CourseModelDS **::** doRetrieveByKey (String code) **Pre:** code !=null and code!="" **Context** CourseModelDS **::** doRetrieveByName(String name)

	Pre: name!=null and name!=”” **Context** CourseModelDS **::** doSave(CourseBean item) **Pre:** item != null
Post-Condizione	**Context** CourseModelDS **::** doRetrieveByKey(String code) **Post:** return Collection <CourseBean> **Context** CourseModelDS **::** doRetrieveByName(String name) **Post:** restituisce il codice del Corso di cui abbiamo inserito il nome, altrimenti -1 **Context** CourseModelDS **::** doSave(CourseBean item) **Post:** viene inserita una nuova istanza nella tabella Corso del database con i valori contenuti in item
Invarianti	

2.1.10 Acquisto Class Interfaces



- **PurchaseBean:** Rappresenta l’entità Acquisto all’interno del database.

Nome Classe	PurchaseModelDS
Descrizione	Questa classe è un manager che si occupa di interagire con il database e permette di interrogare la tabella File

Pre-Condizione	**Context** PurchaseModelDS **::** doRetrieveByUsername (String username) **Pre:** username!=null AND username!="" **Context** PurchaseModelDS **::** doSave (PurchaseBean item) **Pre:** item!=null
Post-Condizione	**Context** PurchaseModelDS **::** doRetrieveByUsername(String username) **Post:** return Collection<PurchaseBean>, una collezione di tutti gli acquisti dell'Utente con Username *username* **Context** PurchaseModelDS **::** doSave (PurchaseBean item) **Post:** viene inserita una nuova istanza nella tabella Acquisto del database con i valori contenuti in item
Invarianti	

3. DESIGN PATTERN

Nel sistema realizzato viene utilizzato un pattern DAO, esso è usato per separare la logica di business dalla logica di accesso ai dati. Infatti, i componenti della logica di business non dovrebbero mai accedere direttamente al database, questo perché, sarebbe sintomo di scarsa manutenibilità.