Università degli Studi di Salerno
Corso di Ingegneria del Software

ClipShot
Object Design Document
Versione 1.2

Data: 10/02/2019

Coordinatore del progetto:

Nome	Matricola
Gilberto Recupito	0512104706

Partecipanti:

Nome	Matricola
Stefano Linguiti	0512104994
Gilberto Recupito	0512104706
Adalgiso Della Calce	0512104796
Carmine Cristian Cruoglio	0512104526

Scritto da:	Tutto Il Team

Revision History

Data	Versione	Descrizione	Autore
28/01	0.1	Aggiunta Trade Offs, Linee Guida Documentazione	Gilberto Recupito
30/01	0.2	Aggiunta Panoramica Packages	Tutto Il Team
2/02	0.3	Aggiunta Interfacce delle Classi	Tutto Il Team
4/02	0.4	Aggiunta Object Constraint Language	Gilberto Recupito
5/02	1.0	Prima Stesura Del Documento	Tutto Il Team
6/02	1.1	Correzione degli errori sull' Object Constraint Language	Tutto il Team
10/02	1.2	Ridefinizione del documento e Ultima Stesura Tutto il Team	Tutto Il Team

1. Introduzione **4**

1.1 Object Design Trade-Off 4

1.2 Linee guida per la Documentazione delle Interfacce 5

1.3 Definizioni,Acronimi e Abbreviazioni 6

1.4 Riferimenti 6

2. Packages **7**

2.1 ClipShot 9

2.2 Descrizione delle Classi 9

2.2.1 Model 9

2.2.2 Control 10

2.2.3 View 13

3. Interfacce delle Classi **17**

4. Design Pattern **35**

1. Introduzione

1.1. Object design trade-offs

Dopo aver stilato il documento di Requirements Analysis e il documento di System Design in cui vi è una descrizione sommaria di ciò che sarà il nostro sistema, definendo i nostri obiettivi ma tralasciando gli aspetti implementativi, andiamo ora a stilare il documento di Object Design che ha come obiettivo quello di produrre un modello che sia in grado di integrare in modo coerente e preciso tutte le funzionalità individuate nelle fasi precedenti.

In particolar modo, in tale documento si definiscono le interfacce delle classi, le operazioni, i tipi, gli argomenti e la signature dei sottosistemi definiti nel System Design. Inoltre sono specificati i trade-off e le linee guida.

Comprensibilità vs Tempo:

Il codice del sistema deve essere comprensibile il più possibile, in modo da facilitare la fase di testing ed eventuali future modifiche da apportare. Per rispettare queste linee guida il codice sarà accompagnato da commenti volti a semplificarne la comprensione. Ovviamente questo comporterà un aumento del tempo di sviluppo del nostro progetto.

Prestazioni vs Costi:

Dato che il nostro progetto è sprovvisto di budget, per poter mantenere prestazioni elevate, in determinate funzionalità verranno utilizzati dei template open source esterni, in particolare Bootstrap.

Interfaccia vs Usabilità:

L'interfaccia grafica è stata realizzata in maniera molto semplice, chiara e concisa, vengono utilizzati i form e pulsanti con lo scopo di rendere semplice l'utilizzo del sistema da parte dell'utente finale.

Sicurezza vs Efficienza:

La sicurezza, come descritto nei requisiti non funzionali del Requirements Analysis,

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. Linee guida per la Documentazione delle Interfacce

Gli sviluppatori dovranno seguire determinate linee guida per la stesura del codice:

Naming Convention:

È buona norma utilizzare nomi:

- Descrittivi
- Pronunciabili
- Di uso comune
- Di lunghezza medio-corta
- Non abbreviati
- Evitando la notazione ungherese
- Utilizzando solo caratteri consentiti (a-z, A-Z, 0-9)

Variabili:

- I nomi delle variabili devono iniziare con la lettera minuscola, e le parole successive con la lettera maiuscola. La dichiarazione delle variabili deve essere effettuata ad inizio blocco; in ogni riga vi deve essere una sola dichiarazione di variabile e va effettuato l'allineamento per migliorare la leggibilità.
- In determinati casi, è possibile utilizzare il carattere underscore "_", ad esempio quando si fa uso di variabili costanti oppure quando si fa uso di proprietà statiche.

Metodi:

- I nomi dei metodi devono iniziare con la lettera minuscola, e le parole successive con la lettera maiuscola. Di solito il nome del metodo è costituito da un verbo che identifica un'azione, seguito dal nome di un oggetto. I nomi dei metodi per l'accesso e la modifica delle variabili dovranno essere del tipo getNomeVariabile() e setNomeVariabile(). Se viene dichiarata una variabile all'interno di un metodo quest'ultima deve essere dichiarata appena prima del suo utilizzo e deve servire per un solo scopo, per facilitare la leggibilità. Esempio: getId(), setId()
- Ai metodi va aggiunta una descrizione, la quale deve essere posizionata prima della dichiarazione del metodo, e deve descriverne lo scopo. La descrizione del metodo deve

includere anche informazioni riguardanti gli argomenti, il valore di ritorno, le eccezioni. I metodi devono essere raggruppati in base alla loro funzionalità.

Classi e pagine:

- I nomi delle classi e delle pagine devono iniziare con la lettera maiuscola, e anche le parole successive all'interno del nome devono iniziare con la lettera maiuscola. I nomi delle classi e delle pagine devono essere evocativi, in modo da fornire informazioni sullo scopo di quest'ultime.
  Ogni file sorgente .java contiene una singola classe e dev'essere strutturato in un determinato modo:
- Una breve introduzione alla classe. L'introduzione indica: l'autore, la versione e la data.

```
/**
* sommario dello scopo della classe.
*
* @author [nome dell'autore]
* @version [numero di versione della classe]
* @since [versione di partenza]
*/
```

- L'istruzione import che permette di importare all'interno della classe gli altri oggetti che la classe utilizza.
- La dichiarazione di una classe è caratterizzata da:
  1. Dichiarazione della classe pubblica
  2. Dichiarazioni di costanti
  3. Dichiarazioni di variabili di classe
  4. Dichiarazioni di variabili d'istanza
  5. Costruttore
  6. Commento e dichiarazione metodi e variabili

1.3. Definizioni, acronimi e abbreviazioni

Acronimi:

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

Abbreviazioni:

- DB: DataBase

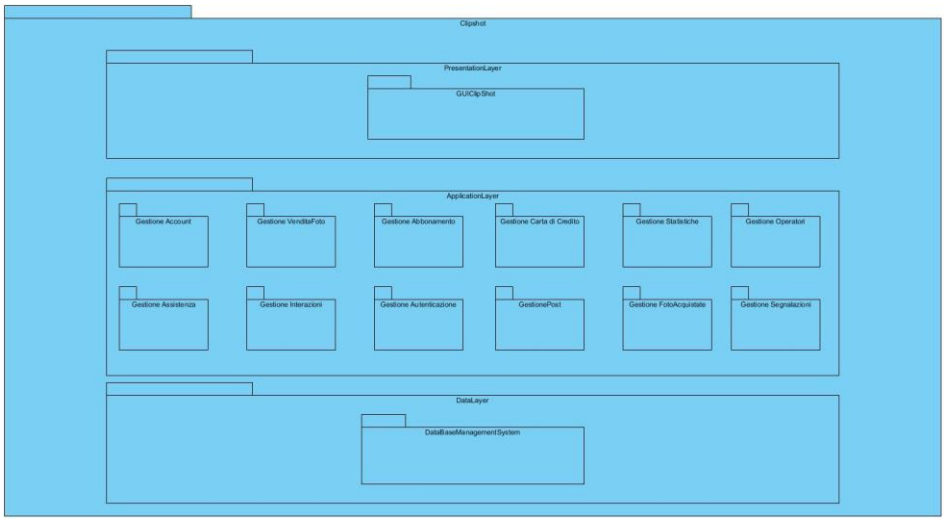
1.4. Riferimenti

- B. Bruegge, A. H. Dutoit, Object Oriented Software Engineering - Using UML, Pattern and Java, Prentice Hall, 3rd edition, 2009
- Documento SDD del progetto ClipShot
- Documento RAD del progetto ClipShot

2. Packages

La struttura del sistema ClipShot è strutturata secondo una divisione in package e sottopackage che raggruppano le classi che hanno il compito di gestirne la logica in base alle richieste dell'utente che ne fa uso.

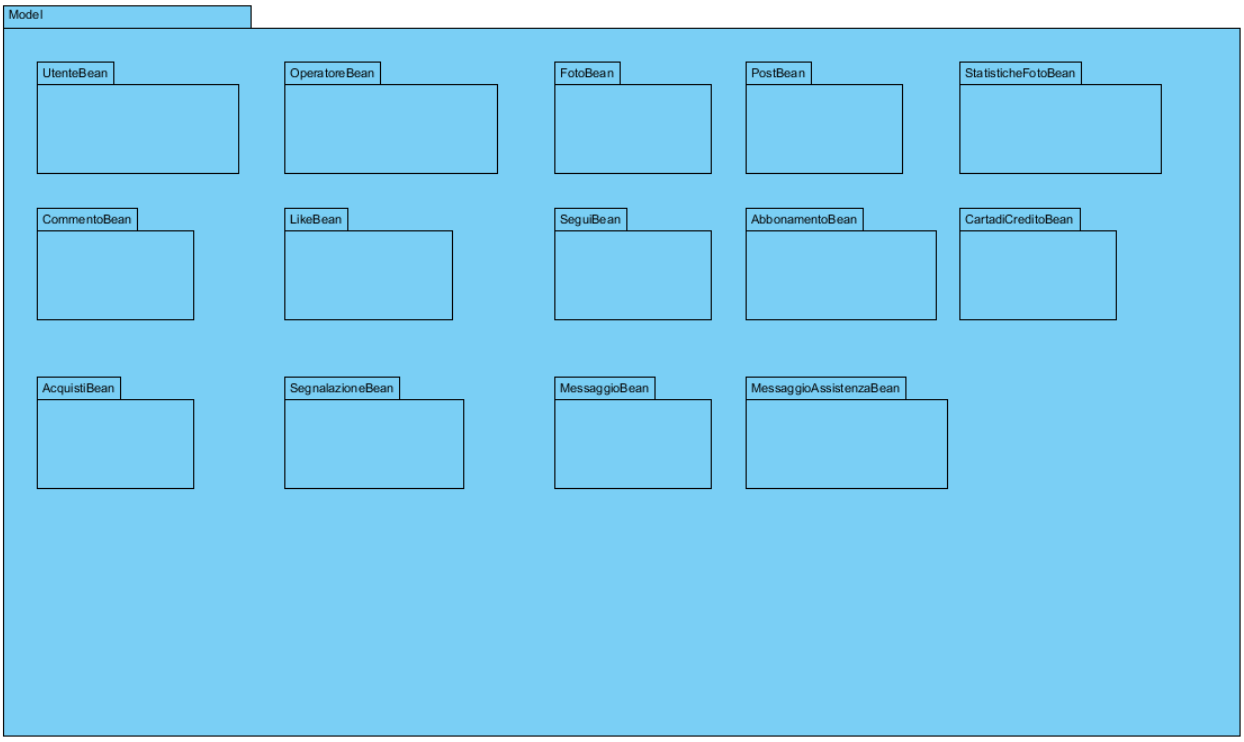
2.1 ClipShot



2.2 Descrizione delle classi

2.1.1 Model

Il sottopackage "Model" è presentato nel seguente schema e contiene le classi java rappresentanti le entità presenti all'interno del sistema.

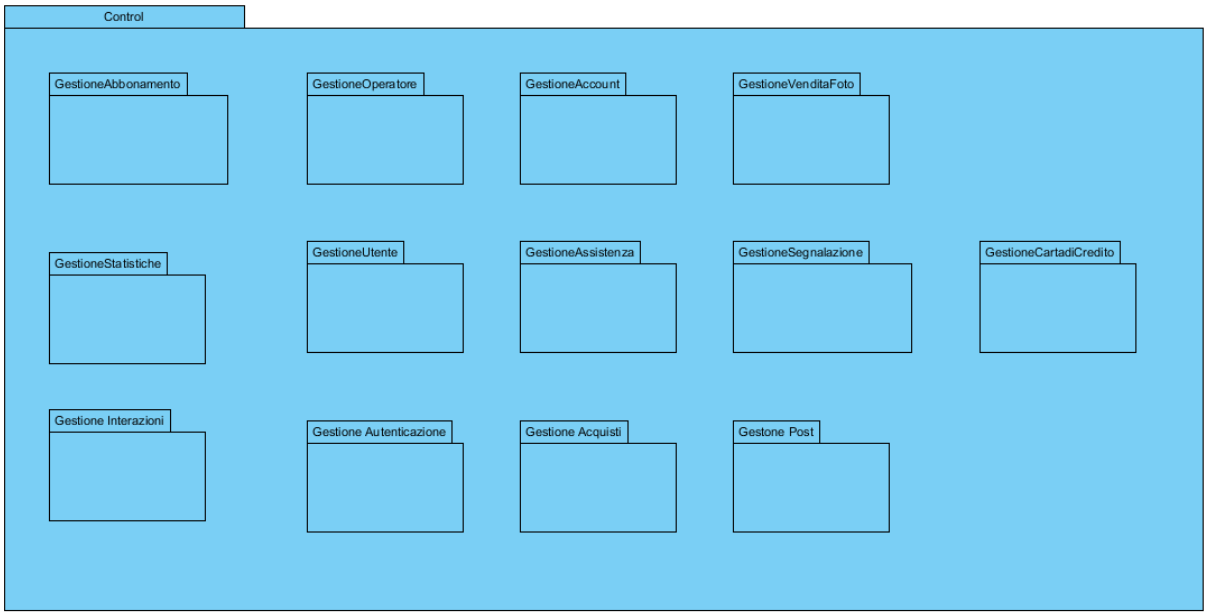


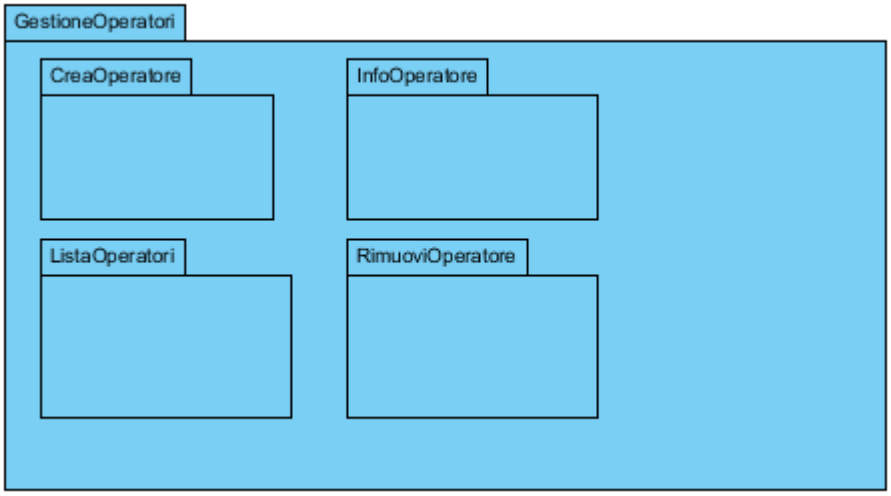
Classe:	**Descrizione:**
UtenteBean	Descrive un utente registrato al sistema.
OperatoreBean	Descrive un operatore.
FotoBean	Descrive una foto.
PostBean	Descrive un post.
StatisticheFotoBean	Descrive le statistiche relative a una fot.
CommentoBean	Descrive il commento di una foto.
LikeBean	Descrive un’interazione “like” di una foto.
SeguiBean	Descrive un’interazione “segui” di un utente.
Abbonamento	Descrive un abbonamento.
AcquistiBean	Descrive un acquisto di una foto.

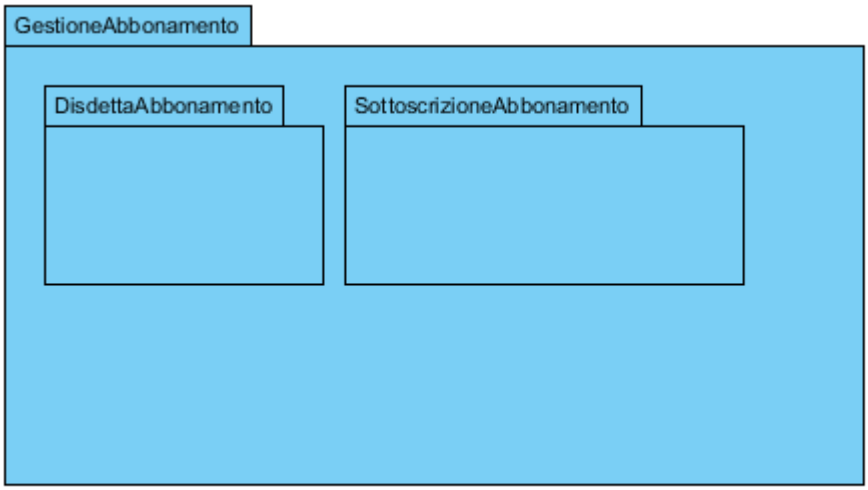
CartadiCredito	Descrive una carta di credito.
SegnalazioneBean	Descrive una segnalazione.
MessaggioBean	Descrive un messaggio.
MessaggioAssistenzaBean	Descrive un messaggio d'assistenza.

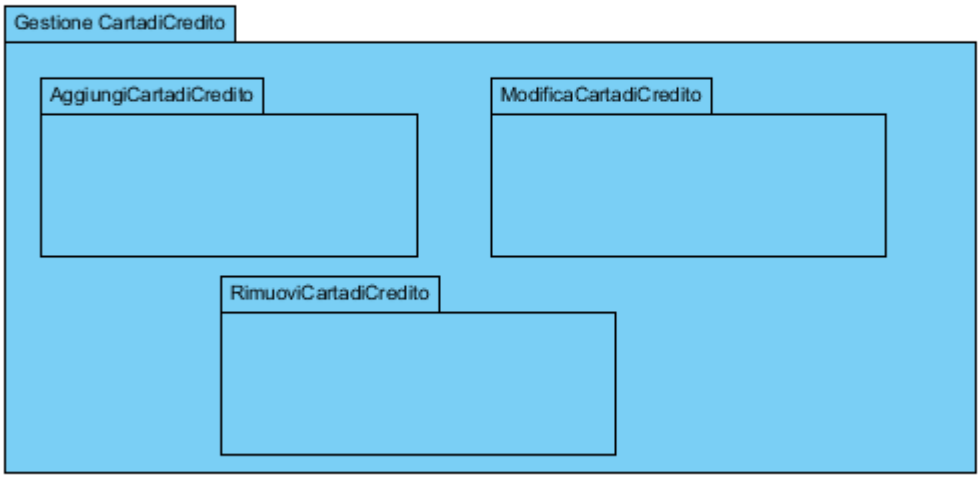
2.2.2 Control

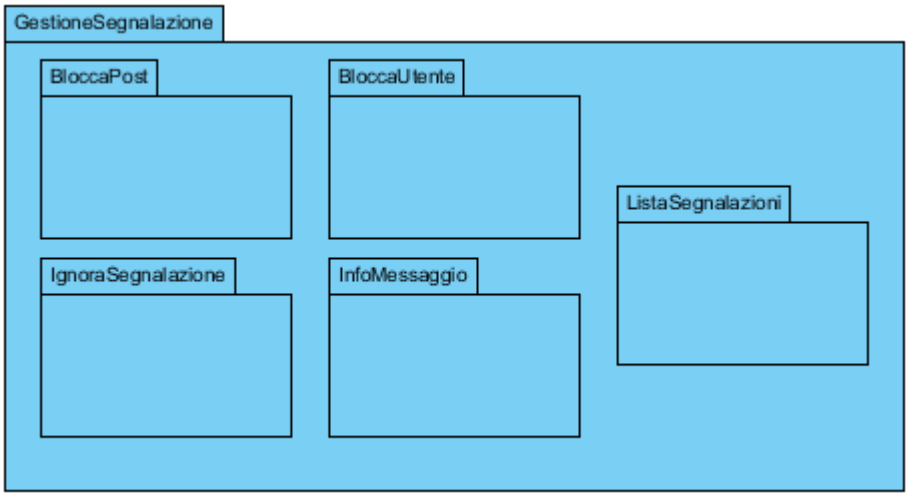
Il sottopackage "Control" è presentato nel seguente schema e contiene le classi Java che si occupano della logica di controllo del sistema.

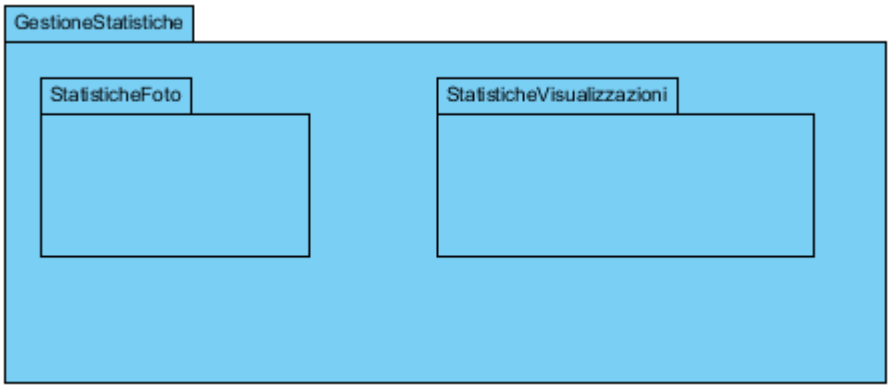


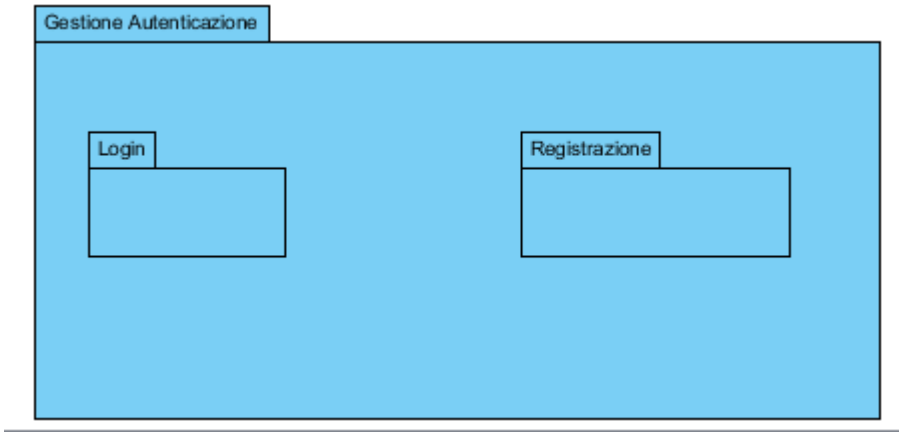

GestioneOperatori
CreaOperatore
InfoOperatore
ListaOperatori
RimuoviOperatore

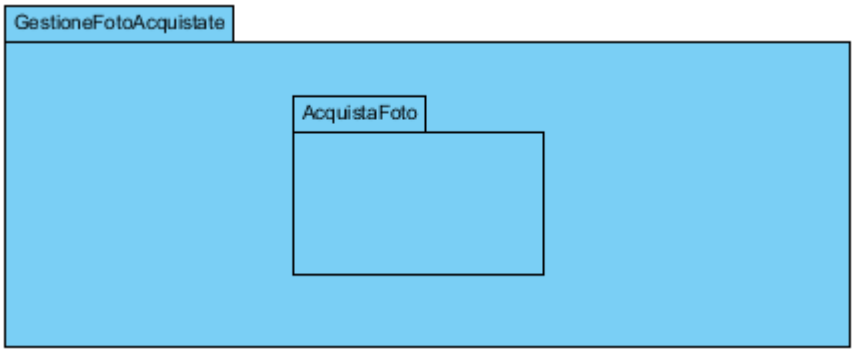

GestioneAbbonamento
DisdettaAbbonamento
SottoscrizioneAbbonamento

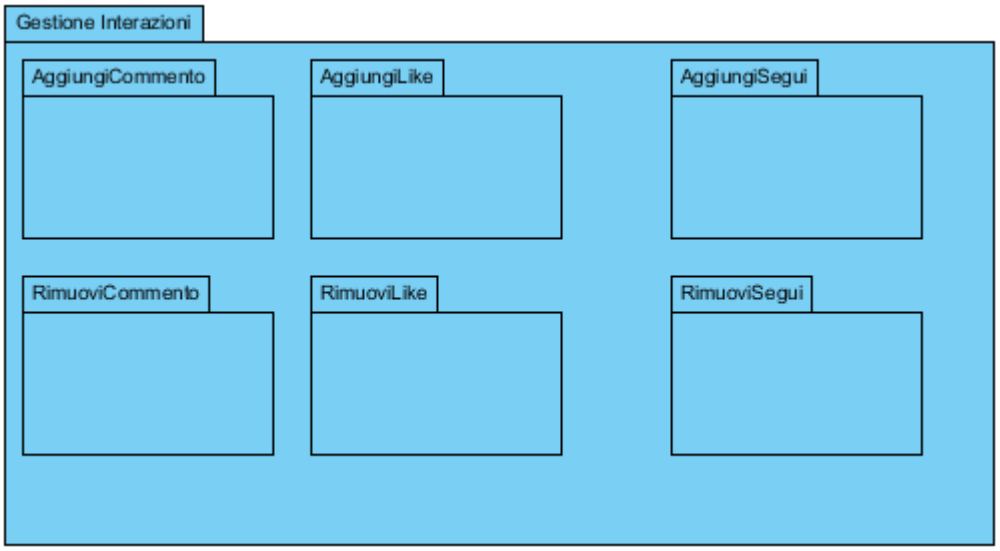

Gestione CartadiCredito
AggiungiCartadiCredito
ModificaCartadiCredito
RimuoviCartadiCredito

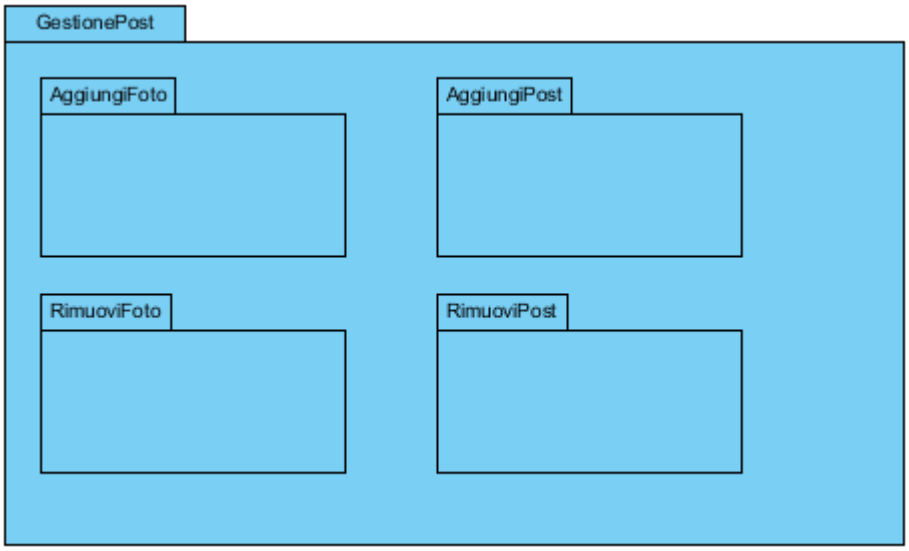

GestioneSegnalazione
BloccaPost
BloccaUtente
ListaSegnalazioni
IgnoraSegnalazione
InfoMessaggio


GestioneStatistiche
StatisticheFoto
StatisticheVisualizzazioni


Gestione Autenticazione
Login
Registrazione

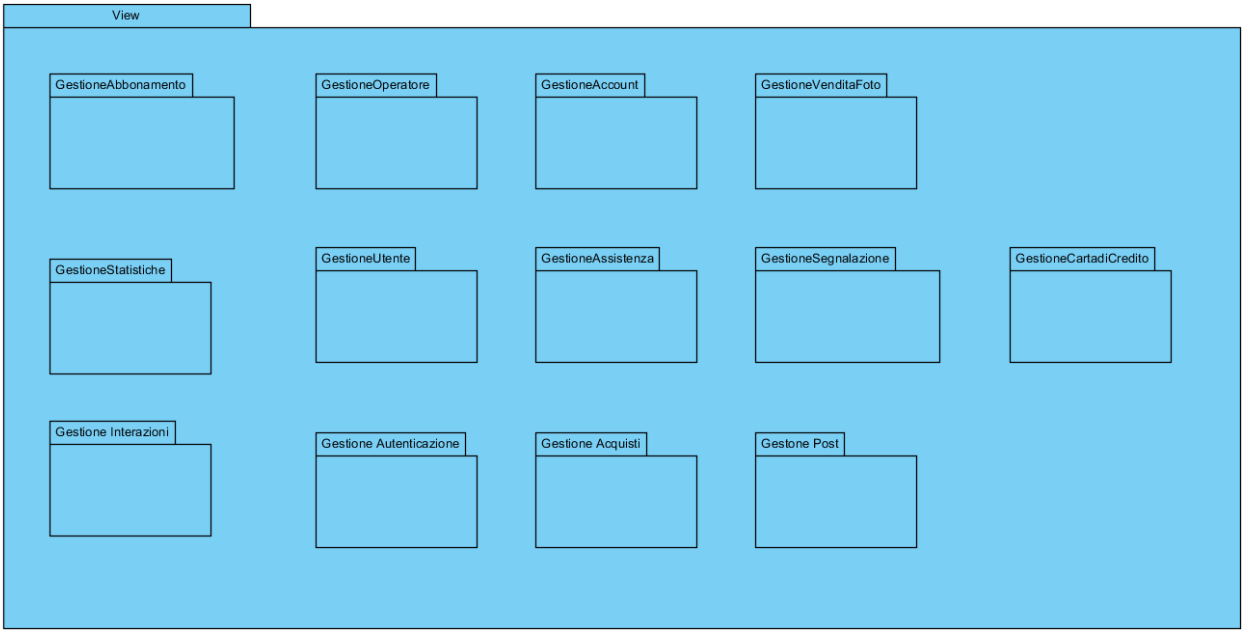

GestioneFotoAcquistate
AcquistaFoto

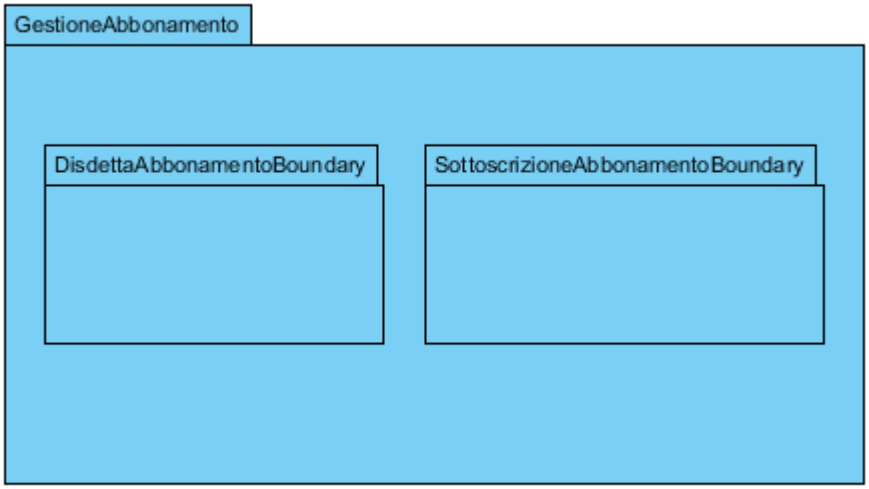

Gestione Interazioni
AggiungiCommento
AggiungiLike
AggiungiSegui
RimuoviCommento
RimuoviLike
RimuoviSegui

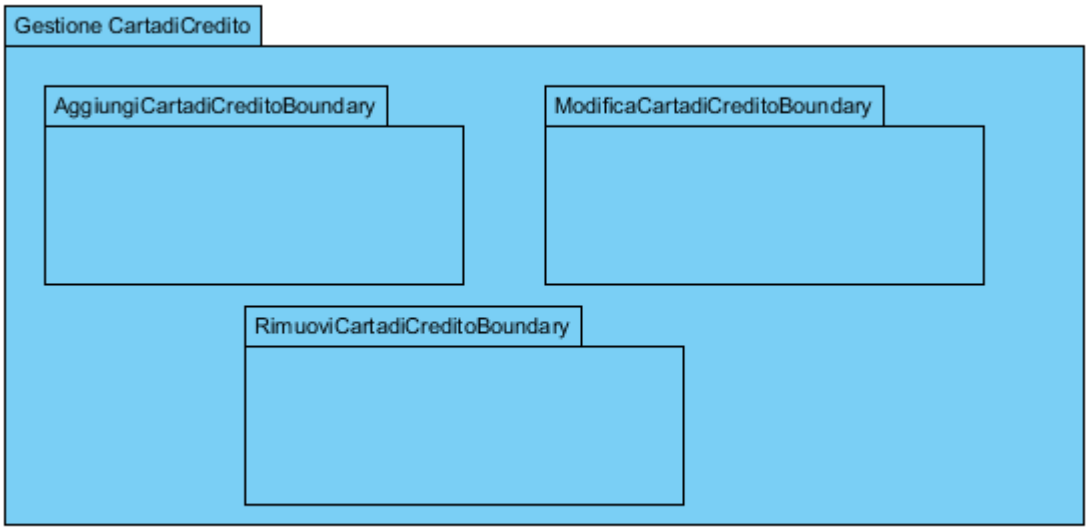


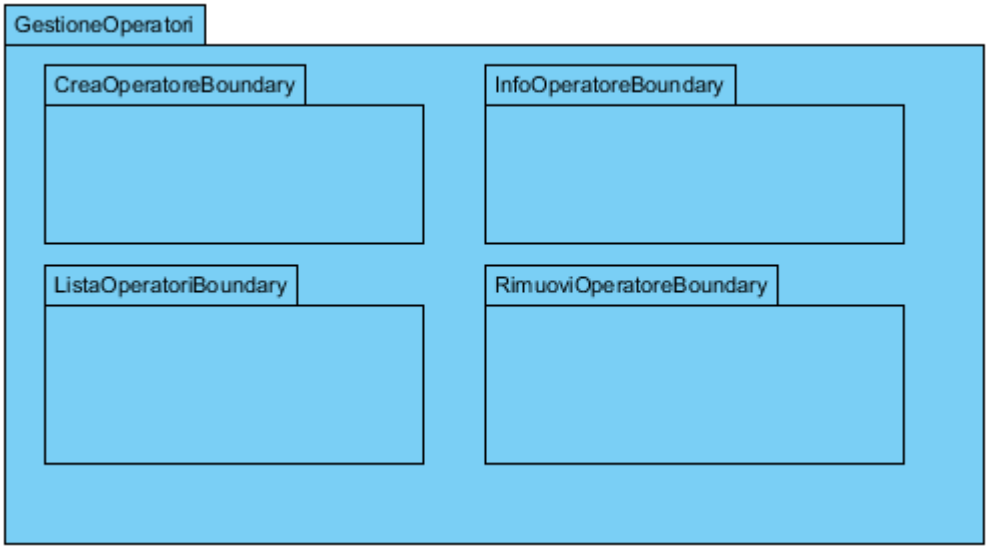
2.2.3 View

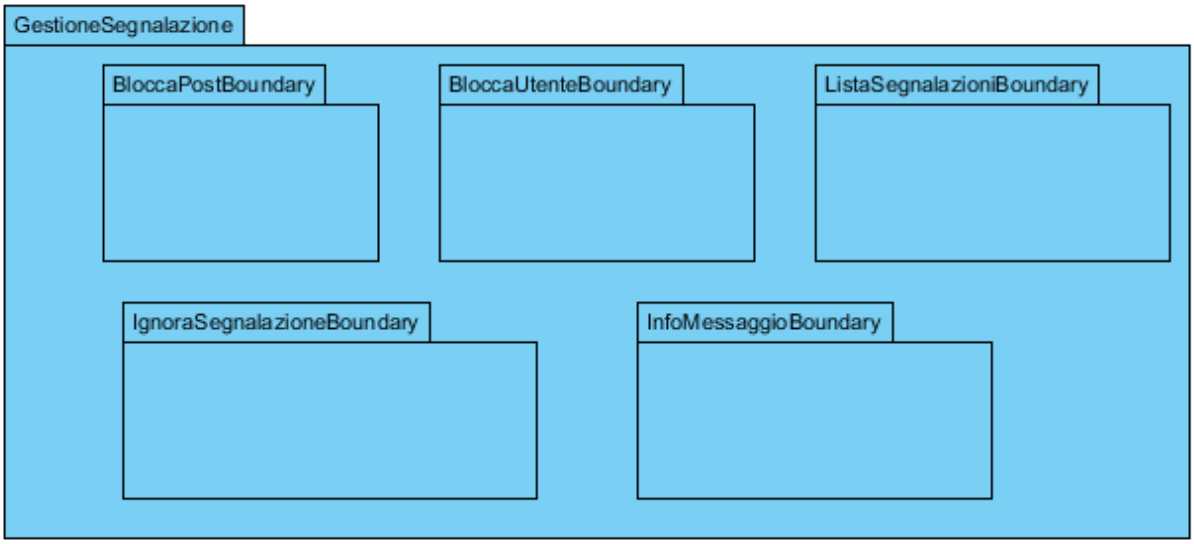
Il sottopackage “View” è presentato nel seguente schema e contiene le classi Java che si occupano della logica di presentazione del sistema.

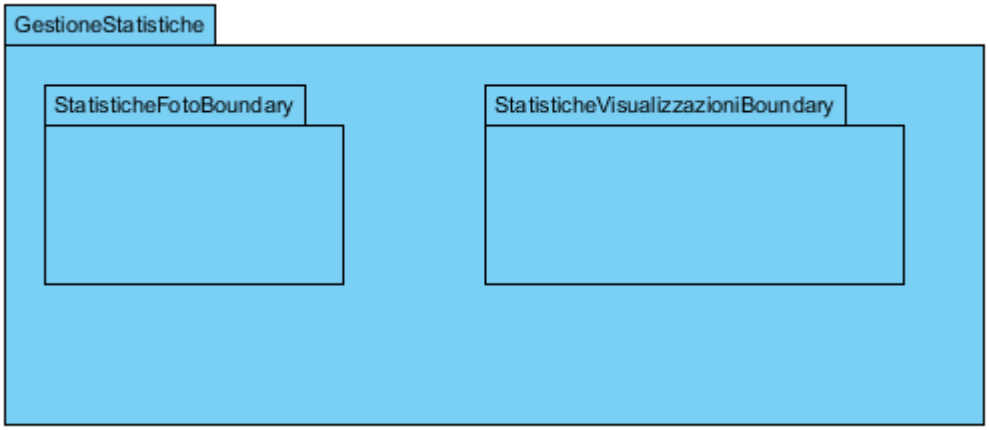


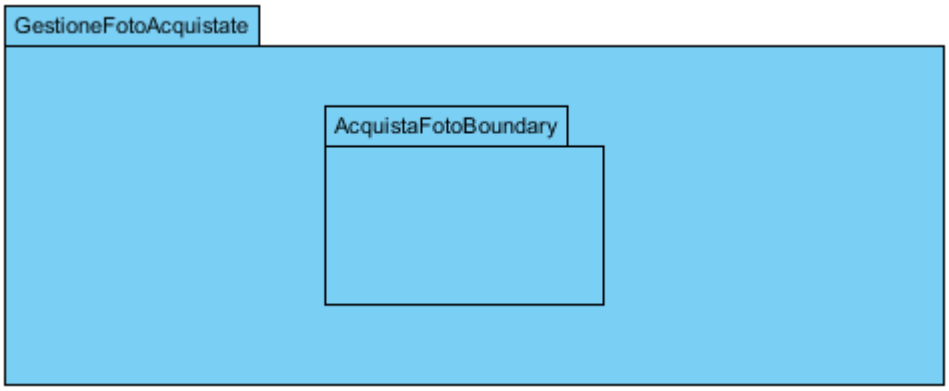

GestioneAbbonamento
DisdettaAbbonamentoBoundary
SottoscrizioneAbbonamentoBoundary

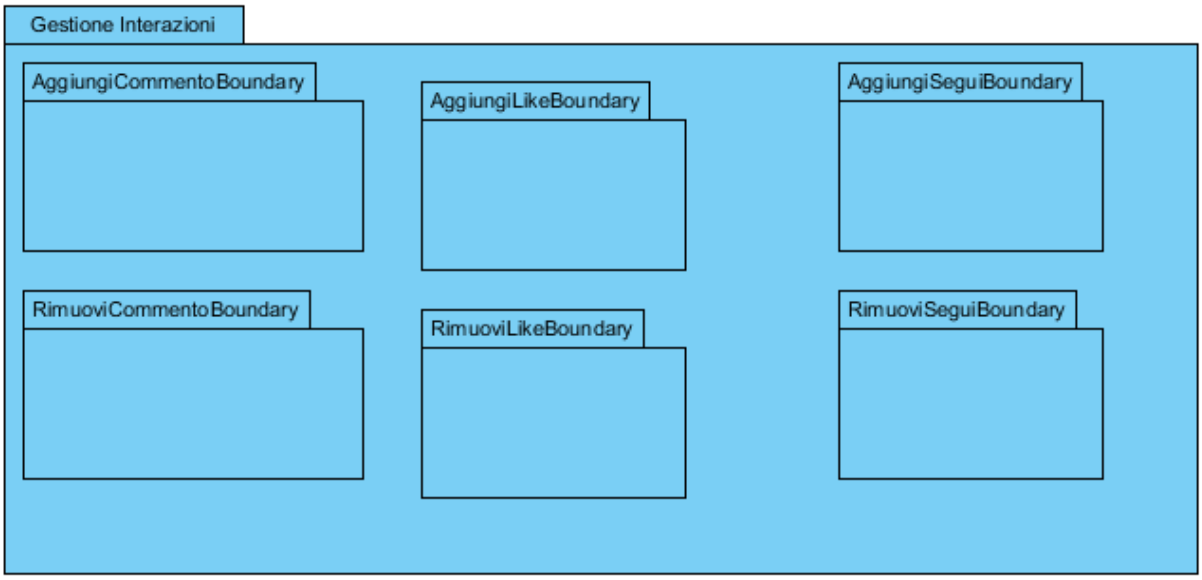

Gestione CartadiCredito
AggiungiCartadiCreditoBoundary
ModificaCartadiCreditoBoundary
RimuoviCartadiCreditoBoundary

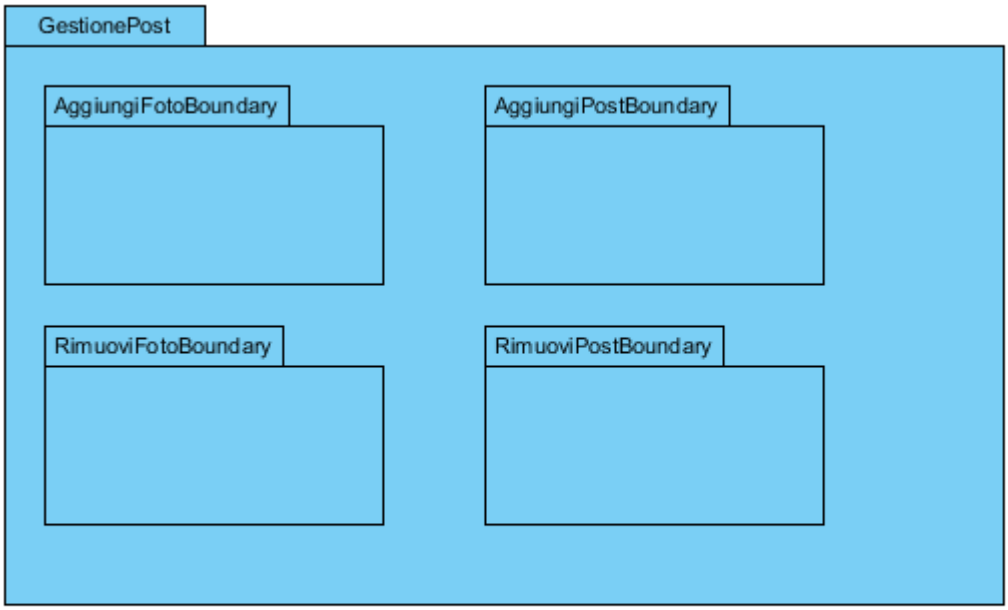

GestioneOperatori
CreaOperatoreBoundary
InfoOperatoreBoundary
ListaOperatoriBoundary
RimuoviOperatoreBoundary


GestioneSegnalazione
BloccaPostBoundary
BloccaUtenteBoundary
ListaSegnalazioniBoundary
IgnoraSegnalazioneBoundary
InfoMessaggioBoundary


GestioneStatistiche
StatisticheFotoBoundary
StatisticheVisualizzazioniBoundary


GestioneFotoAcquistate
AcquistaFotoBoundary


Gestione Interazioni
AggiungiCommentoBoundary
AggiungiLikeBoundary
AggiungiSeguiBoundary
RimuoviCommentoBoundary
RimuoviLikeBoundary
RimuoviSeguiBoundary



3. Interfacce delle Classi

UtenteDAO	
Servizio	**Descrizione**
public void doSave(UtenteBean utente)	Il sottosistema permette di registrare uno utente nel sistema attraverso la compilazione di un apposito form. L'oggetto utente passato come parametro verrà salvato nel database
public void doDelete(UtenteBean utente)	Il sottosistema permette di cancellare un account utente. L' oggetto utente passato come parametro verrà eliminato dal database.
public List<UtenteBean> doRetrieveAll()	Il sottosistema permette di visualizzare i dati degli utenti del sito. La lista di utenti passati come parametro verrà prelevato dal database
public Utente doRetrieveByKey(String idUtente)	Il sottosistema permette di recuperare i dati relativi a un utente registrato nel sito.

	L'idUtente è il codice univoco passato come parametro per prelevare i dati dell' utente dal database.
Public void doSaveorUpdate (Utente utente)	Il sottosistema permette di registrare o di modificare i dati di un cliente registrato nel sistema.

SeguiDAO	
Servizio	**Descrizione**
public void doSave(SeguiBean segui)	Il sottosistema permette di registrare un'interazione "Segui" nel sistema. L'oggetto Segui passato come parametro verrà salvato nel database
public void doDelete(SeguiBean segui)	Il sottosistema permette di cancellare un 'interazione "Segui". L' oggetto Segui passato come parametro verrà eliminato dal database.
public List<SeguiBean> doRetrieveAll()	Il sottosistema permette di visualizzare i dati delle interazioni "Segui" del sito. La lista di interazioni "Segui" passati come parametro verrà prelevato dal database
public SeguiBean doRetrieveByKey(String idFollower, String idFollowing)	Il sottosistema permette di recuperare i dati relativi a un'interazione "Segui" registrata nel sito. L'idFollower e l'idFollowing sono i codici univoci passati come parametri per prelevare i dati dell'interazione "Segui" dal database.

PostDAO	
Servizio	**Descrizione**
public void doSave(PostBean post)	Il sottosistema permette di registrare un Post nel sistema. L'oggetto Post passato come parametro verrà salvato nel database
public void doDelete(PostBean post)	Il sottosistema permette di cancellare un Post. L' oggetto Post passato come parametro verrà eliminato dal database.
public List<PostBean> doRetrieveAll()	Il sottosistema permette di visualizzare i dati di tutti i Post del sito. La lista di Post passati come parametro verrà prelevato dal database

public PostBean doRetrieveByKey(String idPost, String idUtente)	Il sottosistema permette di recuperare i dati relativi a un Post registrata nel sito. L'idPost e l'idUtente sono i codici univoci passati come parametri per prelevare i dati del Post dal database.

LikeDAO	
Servizio	**Descrizione**
public void doSave(LikeBean post)	Il sottosistema permette di registrare un'interazione "Like" nel sistema. L'oggetto Like passato come parametro verrà salvato nel database
public void doDelete(LikeBean post)	Il sottosistema permette di cancellare un 'interazione "Like". L' oggetto Like passato come parametro verrà eliminato dal database.
public List<LikeBean> doRetrieveAll()	Il sottosistema permette di visualizzare i dati delle interazioni "Like" del sito. La lista di interazioni "Like" passati come parametro verrà prelevato dal database.
public LikeBean doRetrieveByKey(String idPost, String idUtente)	Il sottosistema permette di recuperare i dati relativi a un'interazione "Like" registrata nel sito. L'idPost e l'idUtente sono i codici univoci passati come parametri per prelevare i dati dell'interazione "Like" dal database.

FotoDAO	
Servizio	**Descrizione**
public void doSave(FotoBean foto)	Il sottosistema permette di registrare una foto nel sistema. L'oggetto Foto passato come parametro verrà salvato nel database
public void doDelete(FotoBean foto)	Il sottosistema permette di cancellare una foto. L' oggetto Foto passato come parametro verrà eliminato dal database.

public List<FotoBean> doRetrieveAll()	Il sottosistema permette di visualizzare i dati delle Foto del sito. La lista delle Foto passati come parametro verrà prelevato dal database.
public FotoBean doRetrieveByKey(String idFoto)	Il sottosistema permette di recuperare i dati relativi a un Post registrata nel sito. L'idFoto è il codice univoco passato come parametro per prelevare i dati della Foto dal database.

CommentoDAO	
Servizio	**Descrizione**
public void doSave(CommentoBean commento)	Il sottosistema permette di registrare un commento di un post nel sistema. L'oggetto commento passato come parametro verrà salvato nel database
public void doDelete(CommentoBean commento)	Il sottosistema permette di cancellare una foto. L' oggetto Commento passato come parametro verrà eliminato dal database.
public List<CommentoBean> doRetrieveAll()	Il sottosistema permette di visualizzare i dati dei commenti del sito. La lista dei commenti passati come parametro verrà prelevato dal database.
public CommentoBean doRetrieveByKey(String idUtente, String idPost)	Il sottosistema permette di recuperare i dati relativi a una foto registrata nel sito. L'idPost e l'idUtente sono i codici univoci passati come parametri per prelevare i dati del commento dal database.

AcquistiDAO	
Servizio	**Descrizione**
public void doSave(AcquistiBean abbonamento)	Il sottosistema permette di registrare un acquisto di una foto nel sistema. L'oggetto Acquisto passato come parametro verrà salvato nel database
public void doDelete(AcquistiBean abbonamento)	Il sottosistema permette di cancellare una foto.

	L' oggetto Acquisto passato come parametro verrà eliminato dal database.
public List<AcquistiBean> doRetrieveAll()	Il sottosistema permette di visualizzare i dati delle Acquisto del sito. La lista degli Acquisti passati come parametro verrà prelevato dal database.
public AcquistiBean doRetrieveByKey(String idUtente,String idFoto)	Il sottosistema permette di recuperare i dati relativi a un'acquisto registrata nel sito. L'idFoto e l'idUtente sono i codici univoci passati come parametri per prelevare i dati dell'acquisto dal database.

AbbonamentoDAO	
Servizio	**Descrizione**
public void doSave(AbbonamentoBean abbonamento)	Il sottosistema permette di registrare un abbonamento nel sistema. L'oggetto Abbonamento passato come parametro verrà salvato nel database
public void doDelete(AbbonamentoBean abbonamento)	Il sottosistema permette di cancellare una foto. L' oggetto Abbonamenti passato come parametro verrà eliminato dal database.
public List<AbbonamentoBean> doRetrieveAll()	Il sottosistema permette di visualizzare i dati delle Abbonamenti del sito. La lista degli Abbonamenti passati come parametro verrà prelevato dal database.
public AbbonamentoBean doRetrieveByKey(String idAbbonamento)	Il sottosistema permette di recuperare i dati relativi a un abbonamento registrata nel sito. L'idAbbonamento è il codice univoco passati come parametri per prelevare i dati dell'abbonamento dal database.

CartadiCreditoDAO	
Servizio	**Descrizione**
public void doSave(CartadiCredito Bean cartadicredito)	Il sottosistema permette di registrare una Carta di Credito nel sistema. L'oggetto CartadiCredito passato come parametro verrà salvato nel database
public void doDelete(CartadiCredito Bean cartadicredito)	Il sottosistema permette di cancellare una carta di credito. L' oggetto CartadiCredito passato come parametro verrà eliminato dal database.
public List<CartadiCreditoBean> doRetrieveAll()	Il sottosistema permette di visualizzare i dati delle Carte di Credito del sito.

	La lista delle Carte di Credito passati come parametro verrà prelevato dal database.
public CartadiCreditoBean doRetrieveByKey(String ncarta)	Il sottosistema permette di recuperare i dati relativi a una carta di credito registrata nel sito. L'attributo ncarta è il codice univoco passati come parametri per prelevare i dati della Carta di Credito dal database.

MessaggioAssistenzaDAO	
Servizio	**Descrizione**
public void doSave(MessaggioAssistenzaBean messaggioAssistenza)	Il sottosistema permette di registrare un messaggio d'Assistenza nel sistema. L'oggetto MessaggioAssistenza passato come parametro verrà salvato nel database
public void doDelete(MessaggioAssistenzaBean messaggioAssistenza)	Il sottosistema permette di cancellare un messaggio d'assistenza. L' oggetto MessaggioAssistenza passato come parametro verrà eliminato dal database.
public List<MessaggioAssistenzaBean> doRetrieveAll()	Il sottosistema permette di visualizzare i dati dei messaggi d'assistenza del sito. La lista di "MessaggioAssistenza" passati come parametro verrà prelevato dal database.
public MessaggioAssistenzaBean doRetrieveByKey(String idUtente,String idOperatore,Date datamessaggio,Time ora)	Il sottosistema permette di recuperare i dati relativi a un abbonamento registrata nel sito. Gli attributi idUtente, idOperatore, datamessaggio e ora sono i codici univoci passati come parametri per prelevare i dati del Messaggio d'assistenza dal database.

MessaggioDAO	
Servizio	**Descrizione**
public void doSave(MessaggioBean messaggio)	Il sottosistema permette di registrare un messaggio nel sistema. L'oggetto Messaggio passato come parametro verrà salvato nel database
public void doDelete(MessaggioBean messaggio)	Il sottosistema permette di cancellare un messaggio.

	L' oggetto Messaggio passato come parametro verrà eliminato dal database.
public List<MessaggioBean> doRetrieveAll()	Il sottosistema permette di visualizzare i dati dei messaggi del sito. La lista dei messaggi passati come parametro verrà prelevato dal database.
public MessaggioBean doRetrieveByKey(String idMittente,String idDestinatario, Date datamessaggio, Time ora)	Il sottosistema permette di recuperare i dati relativi a un messaggio registrata nel sito. L'attributo idMittente, idDestinatario, datamessaggio e ora sono i codici univoci passati come parametri per prelevare i dati del Messaggio dal database

OperatoreDAO	
Servizio	**Descrizione**
public void doSave(OperatoreBean operatore)	Il sottosistema permette di registrare un operatore nel sistema. L'oggetto Operatore passato come parametro verrà salvato nel database
public void doDelete(OperatoreBean operatore)	Il sottosistema permette di cancellare un operatore. L' oggetto Operatore passato come parametro verrà eliminato dal database.
public List<OperatoreBean> doRetrieveAll()	Il sottosistema permette di visualizzare i dati degli operatori del sito. La lista degli operatori passati come parametro verrà prelevato dal database.
public OperatoreBean doRetrieveByKey(String idOperatore)	Il sottosistema permette di recuperare i dati relativi a un operatore registrato nel sito. L'attributo idOperatore è il codice univoco passato come parametro per prelevare i dati dell'Operatore dal database.

SegnalazioneDAO	
Servizio	**Descrizione**
public void doSave(SegnalazioneBean segnalazione)	Il sottosistema permette di registrare una segnalazione nel sistema. L'oggetto Segnalazione passato come parametro verrà salvato nel database
public void doDelete(SegnalazioneBean segnalazione)	Il sottosistema permette di cancellare una segnalazione.

	L' oggetto Segnalazione passato come parametro verrà eliminato dal database.
public List<SegnalazioneBean> doRetrieveAll()	Il sottosistema permette di visualizzare i dati delle segnalazioni del sito. La lista delle segnalazioni passati come parametro verrà prelevato dal database.
public OperatoreBean doRetrieveByKey(String idUtente,String idSegnalazione)	Il sottosistema permette di recuperare i dati relativi a una segnalazione registrata nel sito. Gli attributi idUtente e id Segnalazioni sono i codici univoci passati come parametri per prelevare i dati della Segnalazione dal database.

StatisticheDAO	
Servizio	**Descrizione**
public void doSave(StatisticheBean statistiche)	Il sottosistema permette di registrare un'insieme di statistiche relative a un utente nel sistema. L'oggetto Statistiche passato come parametro verrà salvato nel database
public void doDelete(StatisticheBean statistiche)	Il sottosistema permette di cancellare le statistiche relative a un utente. L' oggetto Statistiche passato come parametro verrà eliminato dal database.
public List<StatisticheBean> doRetrieveAll()	Il sottosistema permette di visualizzare i dati delle statistiche del sito. La lista delle statistiche passati come parametro verrà prelevato dal database.
public StatisticheBean doRetrieveByKey(String id Utente)	Il sottosistema permette di recuperare i dati relativi alle statistiche di un utente registrate nel sito. L'attributo idUtente è un codice univoco passato come parametro per prelevare i dati della Statistiche dal database.

3.1 Object Constraint Language

Nome classe	UtenteDAO
Pre-condizione	**context** UtenteDAO::public doSave(UtenteBean utente); **pre** String utente.username!= null && String utente.password!=null && String utente.email!=null && String utente.nome!=null && String utente.cognome!=null && String utente.dataNascita!=null && String utente.tipo!=null && String utente.stato!=null **context** UtenteDAO::public doUpdate(UtenteBean utente); **pre** String utente.username!= null && String utente.password!=null && String utente.email!=null && String utente.nome!=null && String utente.cognome!=null && String utente.dataNascita!=null && String utente.tipo!=null && String utente.stato!=null **context** UtenteDAO::public doDelete(String username); **pre** String utente.username!=null **context** UtenteDAO::public doRetrieveByKey(String username); **pre** String username!=null
Post-condizione	**context** UtenteDAO::public doSave(UtenteBean utente); **post** doSave =true **context** UtenteDAO::public doUpdate(Utente utenteBean); **post** doUpdate = true **context** UtenteDAO::public doDelete(String username); **post** doDelete = true **context** UtenteDAO::public doRetrieveByKey(String username); **post** doRetrieveByKey = true
Invarianti	

Nome classe	OperatoreDAO
Pre-condizione	**context** OperatoreDAO::public doSave(OperatoreBean operatore); **pre** String operatore.username!= null && String operatore.password!=null && String operatore.email!=null && String operatore.nome!=null && String operatore.cognome!=null && String operatore.tipo!=null **context** OperatoreDAO::public doUpdate(OperatoreBean operatore); **pre** String operatore.username!= null && String operatore.password!=null && String operatore.email!=null && String

	operatore.nome!=null && String operatore.cognome!=null && String && String operatore.tipo!=null **context** OperatoreDAO::public doDelete(String username); **pre** String operatore.username!=null **context** OperatoreDAO::public doRetrieveByKey(String username); **pre** String operatore.username!=null
Post-condizione	**context** OperatoreDAO::public doSave(OperatoreBean operatore); **post** doSave = true **context** OperatoreDAO::public doUpdate(OperatoreBean operatore); **post** doUpdate = true **context** OperatoreDAO::public doDelete(String username); **post** doDelete = true **context** OperatoreDAO::public doRetrieveByKey(String username); **post** doRetrieveByKey = true
Invarianti	

Nome classe	PostDAO
Pre-condizione	**Context** PostDAO::public doSave(PostBean post); **pre** String post.idPost!= null && String post.descrizione!=null && String post.data!=null && String post.stato!=null && String post.idUtente!=null && String post.idFoto!=null **context** PostDAO::public doUpdate(PostBean post); **pre** String post.idPost!= null && String post.descrizione!=null && Date post.data!=null && String post.stato!=null && String post.idUtente!=null && String post.idFoto!=null **context** PostDAO::public doDelete(String idPost, String username); **pre** String post.idPost!=null && String username !=null **context** PostDAO::public doRetrieveByKey(String idPost,String username); **pre** String post.idPost!=null && String post.username!=null
Post-condizione	**context** PostDAO::public doSave(PostBean post); **post** doSave = true **context** PostDAO::public doUpdate(PostBean post); **post** doUpdate = true **context** PostDAO::public doDelete(String idPost,String username); **post** doDelete = true

	context PostDAO::public doRetrieveByKey(String idPost,String username); **post** doRetrieveByKey = true
Invarianti	

Nome classe	FotoDAO
Pre-condizione	**Context** FotoDAO::public doSave(FotoBean foto); **pre** String foto.idFoto!= null && String foto.path!=null && Double foto.prezzo!=null **context** FotoDAO::public doUpdate(FotoBean post); **pre** int foto.idFoto && String foto.path!=null && Double foto.prezzo !=null **context** FotoDAO::public doDelete(int idFoto); **pre** String foto.idFoto!=null **context** FotoDAO::public doRetrieveByKey(int idFoto); **pre** String foto.idFoto!=null
Post-condizione	**context** FotoDAO::public doSave(FotoBean foto); **post** doSave = true **context** FotoDAO::public doUpdate(FotoBean foto); **post** doUpdate = true **context** FotoDAO::public doDelete(int idFoto); **post** doDelete = true **context** FotoDAO::public doRetrieveByKey(int idFoto); **post** doRetrieveByKey = true
Invarianti	

Nome classe	AbbonamentoDAO
Pre-condizione	**Context** AbbonamentoDAO::public doSave(AbbonamentoBean abbonamento); **pre** String abbonamento.idUtente!= null && Date abbonamento.datascadenza!=null && int abbonamento.numeroCarta!=null && String abbonamento.stato!=null **context** AbbonamentoDAO::public doUpdate(AbbonamentoBean abbonamento); **pre** String abbonamento.idUtente!= null && Date abbonamento.datascadenza!=null && String abbonamento.numeroCarta!=null && String abbonamento.stato!=null **context** AbbonamentoDAO::public doDelete(String idUtente); **pre** String abbonamento.idUtente!=null **context** AbbonamentoDAO::public doRetrieveByKey(String idUtente); **pre** String abbonamento.idUtente!=null
Post-condizione	**context** AbbonamentoDAO::public doSave(AbbonamentoBean abbonamento); **post** doSave = true **context** AbbonamentoDAO::public doUpdate(AbbonamentoBean abbonamento); **post** doUpdate = true **context** AbbonamentoDAO::public doDelete(String idUtente); **post** doDelete = true **context** AbbonamentoDAO::public doRetrieveByKey(String idUtente); **post** doRetrieveByKey = true
Invarianti	

Nome classe	AcquistiDAO
Pre-condizione	**Context** AcquistiDAO::public doSave(AcquistiBean acquisto); **pre** String acquisto.idUtente!= null && Date acquisto.data!=null && int acquisto.idFoto!=null **context** AcquistiDAO::public doUpdate(AcquistiBean acquisti); **pre** String acquisto.idUtente!= null && Date acquisto.idFoto!=null && Date acquisto.data!=null

	context AcquistiDAO::public doDelete(String idUtente); **pre** String acquisto.idUtente!=null **context** AcquistiDAO::public doRetrieveByKey(String idUtente,String idFoto); **pre** String acquisto.idUtente!=null &&String acquisto.idFoto!=null
Post-condizione	**context** AcquistiDAO::public doSave(AcquistiBean acquisto); **post** doSave = true **context** AcquistiDAO::public doUpdate(AcquistiBean acquisto); **post** doUpdate = true 1. **context** AcquistiDAO::public doDelete(String idUtente,String idFoto); 2. **post** doDelete = true 3. **context** AcquistiDAO::public doRetrieveByKey(String idUtente,String idFoto); 4. **post** doRetrieveByKey = true
5. Invarianti	6.

Nome classe	CommentoDAO
Pre-condizione	**Context** CommentoDAO::public doSave(CommentoBean commento); **pre** String commento.idUtente!= null && Date commento.data!=null && int commento.idPost!=null **context** CommentoDAO::public doUpdate(Commento Bean commento); **pre** String commento.idUtente!= null && Date commento.idFoto!=null && Date commento.data!=null **context** CommentoDAO::public doDelete(String idUtente,int idPost, Date data, Date ora); **pre** String commento.idUtente!=null && int commento.idPost!=null && Date commento.data!=null && Date commento.ora !=null **context** CommentoDAO::public doRetrieveByKey(String idUtente,String idPost, Date data, Date ora); **pre** String commento.idUtente!=null && int commento.idPost!=null && Date commento.data!=null && Date commento.ora !=null

Post-condizione	**context** CommentoDAO::public doSave(CommentoBean commento); **post** doSave = true **context** CommentoDAO::public doUpdate(CommentoBean commento); **post** doUpdate = true **context** CommentoDAO::public doDelete(String idUtente,String idPost, Date data, Date ora); **post** doDelete = true **context** CommentoDAO::public doRetrieveByKey(String idUtente,String idPost, Date data, date ora); **post** doRetrieveByKey = true
Invarianti	

Nome classe	LikeDAO
Pre-condizione	**Context** LikeDAO::public doSave(LikeBean like); **pre** String like.idUtente!= null && Date like.data!=null && int like.idPost!=null **context** LikeDAO::public doUpdate(LikeBean like); **pre** String like.idUtente!= null && Date like.idFoto!=null && Date like.data!=null **context** LikeDAO::public doDelete(String idUtente,int idPost); **pre** String like.idUtente!=null && int like.idPost!=null **context** LikeDAO::public doRetrieveByKey(String idUtente,String idPost); **pre** String like.idUtente!=null && int like.idPost!=null
Post-condizione	**context** LikeDAO::public doSave(LikeBean like); **post** doSave = true **context** LikeDAO::public doUpdate(LikeBean like); **post** doUpdate = true **context** LikeDAO::public doDelete(String idUtente,String idPost, Date data, Date ora); **post** doDelete = true **context** LikeDAO::public doRetrieveByKey(String idUtente,String idPost, Date data, date ora); **post** doRetrieveByKey = true
Invarianti	

Nome classe	CartadiCreditoDAO
Pre-condizione	**Context** CartadiCreditoDAO::public doSave(CartadiCreditoBean carta); **pre** String carta.idUtente!= null && int carta.numeroCarta!=null && String carta.intestatario!=null && Date carta.datascadenza!=null && int carta.cvv !=null; **context** CartadiCreditoDAO::public doUpdate(CartadiCreditoBean carta) **pre** String carta.idUtente!= null && int carta.numeroCarta!=null && String carta.intestatario!=null && Date carta.datascadenza!=null; && int carta.cvv !=null; **context** CartadiCreditoAO::public doDelete(int numerocarta); **pre** String numerocarta!=null; **context** CartadiCreditoDAO::public doRetrieveByKey(String numerocarta); **pre** String numerocarta!=null;
Post-condizione	**context** CartadiCreditoDAO::public doSave(CartadiCreditoBean commento); **post** doSave = true **context** CartadiCreditoDAO::public doUpdate(CartadiCreditoBean commento); **post** doUpdate = true **context** CartadiCreditoDAO::public doDelete(String numerocarta); **context** CartadiCreditoDAO::public doRetrieveByKey(String numerocarta); **post** doRetrieveByKey = true
Invarianti	

Nome classe	MessaggioAssistenzaDAO
Pre-condizione	**Context** MessaggioAssistenzaDAO::public doSave(MessaggioAssistenzaBean messaggio); **pre** String messaggio.idOperatore!= null && String messaggio.idUtente!=null && Date messaggio.data!=null && Date messaggio.ora!=null && String messaggio.oggetto !=null && String messaggio.tipo!=null && String messaggio.corpo!=null;

	context MessaggioAssistenzaDAO::public doUpdate(MessaggioAssistenzaBean messaggio) **pre** String messaggio.idOperatore!= null && String messaggio.idUtente!=null && Date messaggio.data!=null && Date messaggio.ora!=null && String messaggio.oggetto !=null && String messaggio.tipo!=null && String messaggio.corpo!=null; **context** MessaggioAssistenzaDAO::public doDelete(String idOperatore,String idUtente,Date data, Date ora); **pre** String messaggio.idOperatore!= null && String messaggio.idUtente!=null && Date messaggio.data!=null && Date messaggio.ora!=null **context** MessaggioAssistenzaDAO::public doRetrieveByKey(String idOperatore,String idUtente,Date data, Date ora); **pre** String messaggio.idOperatore!= null && String messaggio.idUtente!=null && Date messaggio.data!=null && Date messaggio.ora!=null;
Post-condizione	**context** MessaggioAssistenzaDAO::public doSave(MessaggioAssistenzaBean messaggio); **post** doSave = true **context** MessaggioAssistenzaDAO::public doUpdate(MessaggioAssistenzaBean messaggio); **post** doUpdate = true **context** MessaggioAssistenzaDAO::public doDelete(String idOperatore,String idUtente,Date data, Date ora); **context** MessaggioAssistenzaDAO::public doRetrieveByKey(String idOperatore,String idUtente,Date data, Date ora); **post** doRetrieveByKey = true
Invarianti	

Nome classe	MessaggioDAO
Pre-condizione	**Context** MessaggioDAO::public doSave(MessaggioBean messaggio); **pre** String messaggio.idMittente!= null && String messaggio.idDestinatario!=null && Date messaggio.data!=null && Date messaggio.ora!=null && String messaggio.corpo!=null; **context** MessaggioDAO::public doUpdate(MessaggioBean messaggio) **pre** String messaggio.idMittente!= null && String messaggio.idDestinatario!=null && Date messaggio.data!=null && Date messaggio.ora!=null && String messaggio.corpo!=null;

	context MessaggioDAO::public doDelete(String idMittente,String idDestinatario,Date data, Date ora); **pre** String idMittente!= null && String idDestinatario!=null && Date data!=null && Date ora!=null; **context** MessaggioDAO::public doRetrieveByKey(String idMittente,String idDestinatario,Date data, Date ora); **pre** String idMittente!= null && String idDestinatario!=null && Date data!=null && Date ora!=null;
Post-condizione	**context** MessaggioDAO::public doSave(MessaggioAssistenzaBean messaggio); **post** doSave = true **context** MessaggioDAO::public doUpdate(MessaggioAssistenzaBean messaggio); **post** doUpdate = true **context** MessaggioDAO::public doDelete(String idMittente,String idDestinatario,Date data, Date ora); **context** MessaggioDAO::public doRetrieveByKey(String idMittente,String idDestinatario,Date data, Date ora); **post** doRetrieveByKey = true
Invarianti	

Nome classe	SegnalazioneDAO
Pre-condizione	**Context** SegnalazioneDAO::public doSave(SegnalazioneBean segnalazione); **pre** int segnalazione.idSegnalazione!= null && String segnalazione.idUtente!=null && int segnalazione.idPost!=null && String segnalazione.idUtentePost && String segnalazione.causa !=null && String segnalazione.stato!=null && Date segnalazione.data !=null && String segnalazione.descrizione !=null; **context** SegnalazioneDAO::public doUpdate(SegnalazioneBean messaggio) **pre** int segnalazione.idSegnalazione!= null && String segnalazione.idUtente!=null && int segnalazione.idPost!=null && String segnalazione.idUtentePost && String segnalazione.causa !=null && String segnalazione.stato!=null && Date segnalazione.data !=null && String segnalazione.descrizione !=null;

	context SegnalazioneDAO::public doDelete(String idSegnalazione,String idPost,String idUtente, String idUtentePost); **pre** String idSegnalazione !=null && String idPost!=null &&String idUtente !=null && String idUtentePost!=null ; **context** SegnalazioneDAO::public doRetrieveByKey(String idSegnalazione,String idPost,String idUtente, String idUtentePost); **pre** String idSegnalazione !=null && String idPost!=null &&String idUtente !=null && String idUtentePost!=null ;
Post-condizione	**context** SegnalazioneDAO::public doSave(SegnalazioneBean segnalazione); **post** doSave = true **context** SegnalazioneDAO::public doUpdate(SegnalazioneBean segnalazione); **post** doUpdate = true **context** SegnalazioneDAO::public doDelete(String idSegnalazione,String idPost,String idUtente, String idUtentePost); **context** SegnalazioneDAO::public doRetrieveByKey(String idSegnalazione,String idPost,String idUtente, String idUtentePost); **post** doRetrieveByKey = true
Invarianti	

Nome classe	SeguiDAO
Pre-condizione	**Context** SeguiDAO::public doSave(SeguiBean segui); **pre** String segui.idFollower!= null && String segui.idFollowing!=null ; **context** SeguiDAO::public doUpdate(SeguiBean segui) **pre** String segui.idFollower!= null && String segui.idFollowing!=null ; **context** SeguiDAO::public doDelete(String idFollower, String idFollowing); **pre** String idFollower!= null && String idFollowing!=null ; **context** SeguiDAO::public doRetrieveByKey(String idFollower, String idFollowing); **pre** String idFollower!= null && String idFollowing!=null ;

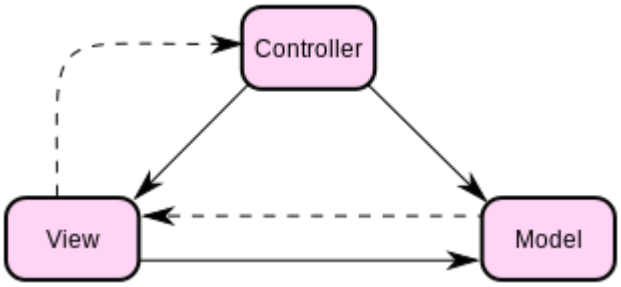
Post-condizione	**context** SeguiDAO::public doSave(SeguiBean segui); **post** doSave = true **context** SeguiDAO::public doUpdate(SeguiBean segui); **post** doUpdate = true **context** SeguiDAO::public doDelete(String idFollower, String idFollowing); **context** SeguiDAO::public doRetrieveByKey(String idFollower, String idFollowing); **post** doRetrieveByKey = true
Invarianti	

Nome classe	StatisticheDAO
Pre-condizione	**Context** StatisticheDAO::public doSave(StatisticheBean statistiche); **pre** String segui.idUtente!= null && int segui.numeroVisualizzazioni!=null ; **context** StatisticheDAO::public doUpdate(StatisticheBean statistiche) **pre** String segui.idUtente!= null && int segui.numeroVisualizzazioni!=null ; **context** StatisticheDAO::public doDelete(String idUtente); **pre** String idUtente!= null **context** StatisticheDAO::public doRetrieveByKey(String idUtente); **pre** String idUtente!= null
Post-condizione	**context** StatisticheDAO::public doSave(StatisticheBean statistiche); **post** doSave = true **context** StatisticheDAO::public doUpdate(StatisticheBean statistiche); **post** doUpdate = true **context** StatisticheDAO::public doDelete(String idFollower, String idFollowing); **context** StatisticheDAO::public doRetrieveByKey(String idFollower, String idFollowing); **post** doRetrieveByKey = true
Invarianti	

4.Design Pattern

Model-View-Controller

Il pattern architetturale Model-View-Controller(MVC) è molto diffuso nello sviluppo di un sistema software, e permette la separazione della logica di business dalla logica di presentazione dei dati.



- Il model fornisce i metodi per accedere ai dati utili all'applicazione;
- Il view visualizza i dati contenuti nel model e si occupa dell'interazione degli utenti con il sistema.
- Il controller riceve i comandi dell'utente (in genere attraverso il view) e li attua modificando lo stato degli altri due componenti.

Proxy Pattern

Il proxy pattern nasce per risolvere problemi relativi al collegamento di un oggetto che ha bisogno di tempi notevoli.

Esso permette la creazione di un oggetto denominato "Proxy" che sostituisce l'oggetto reale.

Tra i vantaggi dell'utilizzo del proxy pattern abbiamo:

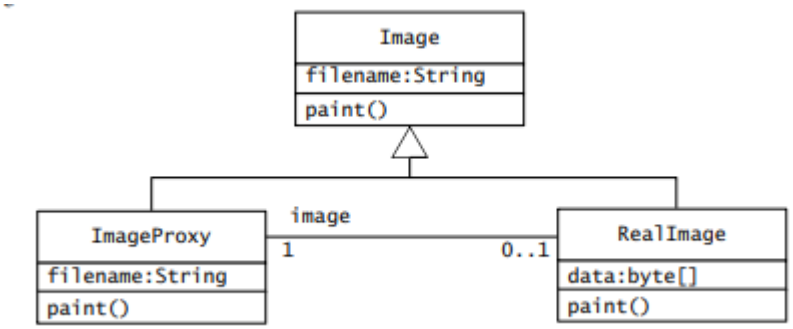
-Alleggerire il carico applicativo: Se il file non è mai necessario, il caricamento risulta essere completamente eliminato.

-Prevede errori nel caso in cui l'oggetto non è disponibile.

Consideriamo quindi per il nostro progetto un oggetto che rappresenta un immagine memorizzata come file. Caricare l'immagine a massima risoluzione è costoso. Quindi possiamo realizzare un Proxy Pattern per risolvere il problema.

L'oggetto ImageProxy prende il posto dell' oggetto Image e utilizza la stessa interfaccia Image. Quando l'Image deve essere scaricata, ImageProxy scarica i dati dal server a crea un RealImage Object. Se invece l'utente non ha bisogno di

quell'immagine , e quindi non invoca il metodo paint(), l'oggetto Real Image non viene creato.



Data-Access-Object

Il *DAO* (*Data Access Object*) è un pattern architetturale per la gestione della persistenza. Si tratta di una classe usata principalmente in applicazioni web , per stratificare e isolare l'accesso ad una tabella tramite query (poste all'interno dei metodi della classe) ovvero al *data layer* da parte della *business logic* creando un maggiore livello di astrazione ed una più facile manutenibilità. I metodi del DAO con le rispettive query dentro verranno così richiamati dalle classi della business logic.

Il vantaggio relativo all'uso del DAO è dunque il mantenimento di una rigida separazione tra le componenti di un'applicazione, le quali potrebbero essere il "Modello" e il "Controllo" in un'applicazione basata sul paradigma MVC.