ODD
Object
Design
Document

Simplify3D

Nicola Librera
Carmine Buondonno
Alessandro Oliviero

Data	Versione	Cambiamenti	Autori
12/12/2019	1.01	Prima stesura.	Nicola Librera Alessandro Oliviero Carmine Buondonno
13/12/2019	1.02	Aggiunta trade-offs e completamento introduzione.	Nicola Librera Alessandro Oliviero Carmine Buondonno
16/12/2019	1.03	Aggiunta di alcuni package diagram.	Nicola Librera Alessandro Oliviero Carmine Buondonno
17/12/2019	1.04	Completamento dei package diagram e inizio interfaccia delle classi.	Nicola Librera Alessandro Oliviero Carmine Buondonno
18/12/2019	1.05	Modifica di design pattern.	Nicola Librera Alessandro Oliviero Carmine Buondonno
10/01/2020	1.06	Revisione documento, modifica dei design pattern e interfaccia delle classi.	Nicola Librera Alessandro Oliviero Carmine Buondonno

Sommario

1. Introduzione

1.1 Trade offs

1.1.1 Comprensibilità vs costi

Si preferisce aumentare i costi per la documentazione al fine di rendere il codice comprensibile anche alle persone non coinvolte nel progetto o le persone coinvolte che non hanno lavorato a quella parte in particolare. Commenti diffusi nel codice facilitano la comprensione, di conseguenza migliorare la comprensibilità agevola il mantenimento e anche il processo di modifica.

1.1.2 Memoria vs efficienza

Si è preferita la memoria anziché le prestazioni perché salvare risultati parziali o complessivi avrebbe causato il mantenimento di informazioni incompleti, difficili da integrare e interpretare.

1.1.3 Comprare vs costruire/gestire

Si preferisce l'acquisto di VPS (Virtual Private Server) piuttosto che la creazione e la gestione di nuovi server all'interno delle infrastrutture universitarie in quanto i VPS messi a disposizioni da numerose aziende di web-services offrono sistemi già preconfigurati, multifunzione ed espandibili.

1.2 Componenti off-the-shelf

Per il progetto software che si vuole realizzare facciamo uso di COTS (Component off-the-shelf) che sono componenti software e hardware già disponibili sul mercato pronte all'utilizzo.
La COTS di tipo hardware utilizzata in tale progetto è un VPS di ultima generazione utilizzato per l'installazione del nostro applicativo. All'interno di questo progetto sono state anche utilizzate COTS di tipo software, fondamentali per lo sviluppo di tale applicativo in quanto già testate prima di essere messe sul mercato. Per il sistema che si vuole realizzare faremo quindi uso di una libreria Java per effettuare l'hashing della password, in particolare si utilizzerà l'algoritmo MD5 e per quanto riguarda la registrazione e il reset delle password è stata utilizzata una libreria di java chiamata mail.jar per l'invio di email agli indirizzi specificati. Infine per consentire l'accesso e la gestione della persistenza dei dati è stato utilizzato il driver JDBC.

1.3 Linee guida per la documentazione dell'interfaccia e convenzioni

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. Le linee guida utilizzate per lo sviluppo del codice sorgente e delle pagine web sono quelle di Google Style Guide le quali possono essere reperiti ai seguenti link:

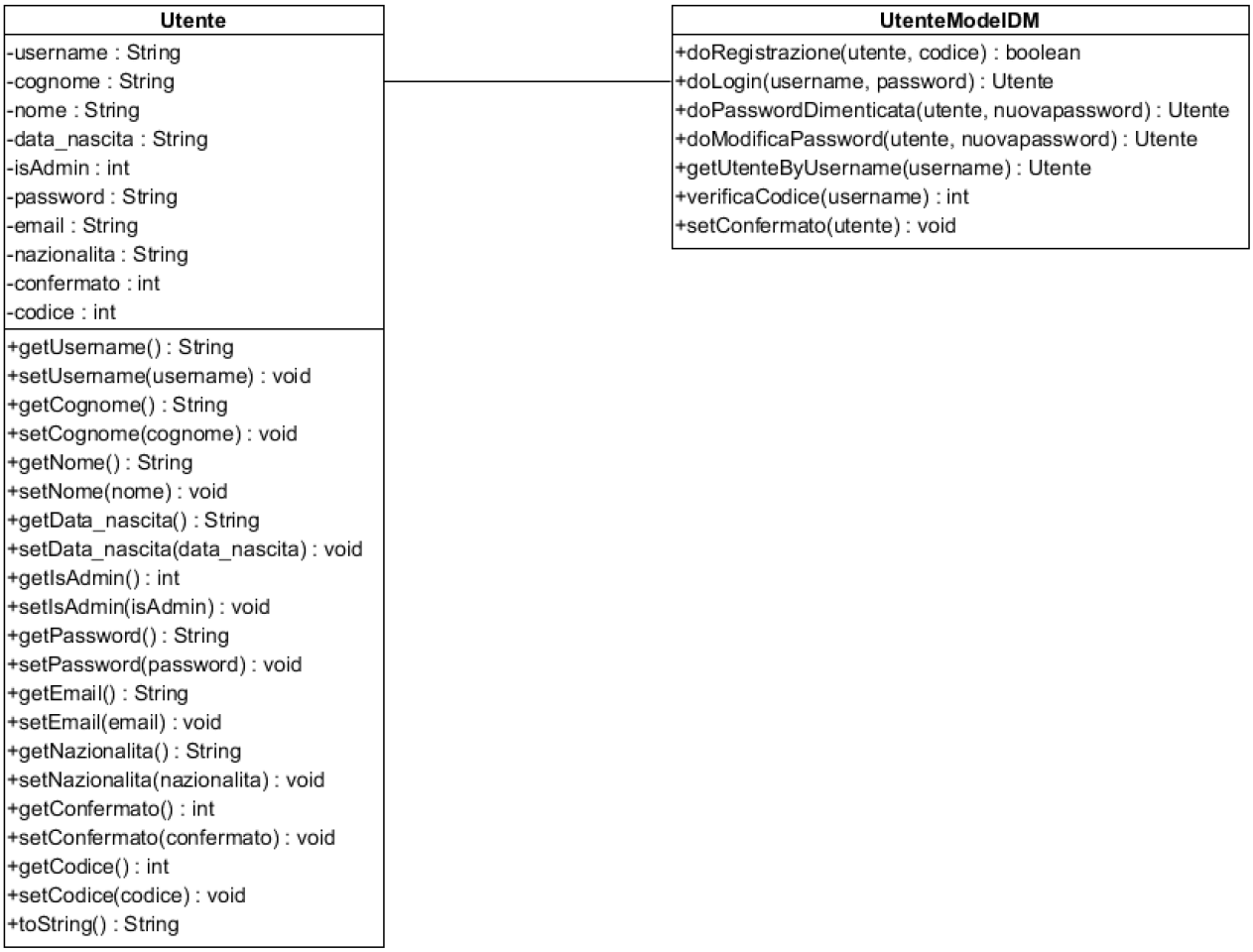
Sviluppo codice sorgente:

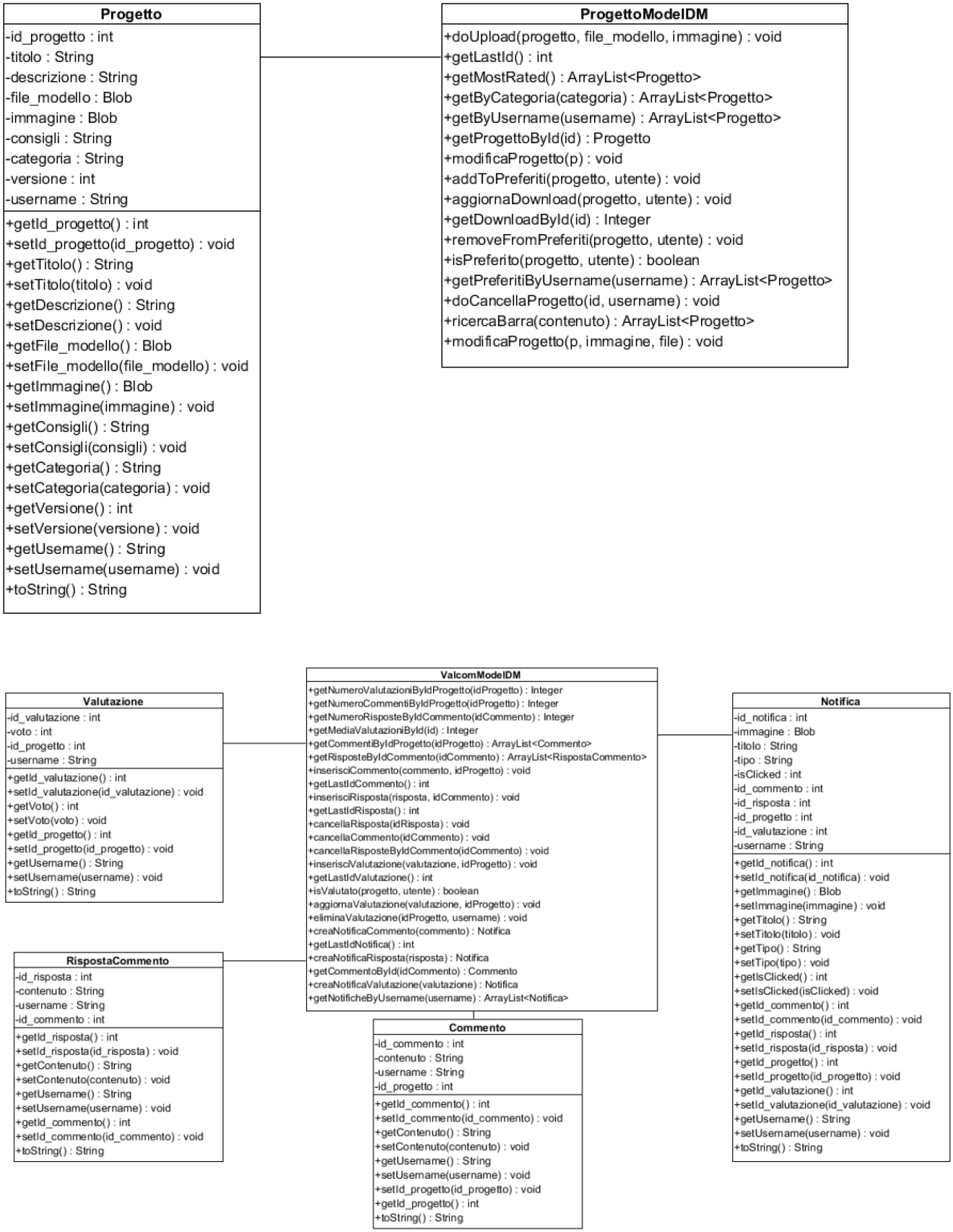
- http://checkstyle.sourceforge.net/reports/google-java-style-20170228.html

Sviluppo pagine web:

- https://google.github.io/styleguide/htmlcssguide.html

Dipendenze dei sottosistemi




Progetto
-id_progetto : int
-titolo : String
-descrizione : String
-file_modello : Blob
-immagine : Blob
-consigli : String
-categoria : String
-versione : int
-username : String
+getId_progetto() : int
+setId_progetto(id_progetto) : void
+getTitolo() : String
+setTitolo(titolo) : void
+getDescrizione() : String
+setDescrizione() : void
+getFile_modello() : Blob
+setFile_modello(file_modello) : void
+getImmagine() : Blob
+setImmagine(immagine) : void
+getConsigli() : String
+setConsigli(consigli) : void
+getCategoria() : String
+setCategoria(categoria) : void
+getVersione() : int
+setVersione(versione) : void
+getUsername() : String
+setUsername(username) : void
+toString() : String
ProgettoModelDM
+doUpload(progetto, file_modello, immagine) : void
+getLastId() : int
+getMostRated() : ArrayList<Progetto>
+getByCategoria(categoria) : ArrayList<Progetto>
+getByUsername(username) : ArrayList<Progetto>
+getProgettoById(id) : Progetto
+modificaProgetto(p) : void
+addToPreferiti(progetto, utente) : void
+aggiornaDownload(progetto, utente) : void
+getDownloadById(id) : Integer
+removeFromPreferiti(progetto, utente) : void
+isPreferito(progetto, utente) : boolean
+getPreferitiByUsername(username) : ArrayList<Progetto>
+doCancellaProgetto(id, username) : void
+ricercaBarra(contenuto) : ArrayList<Progetto>
+modificaProgetto(p, immagine, file) : void
Valutazione
-id_valutazione : int
-voto : int
-id_progetto : int
-username : String
+getId_valutazione() : int
+setId_valutazione(id_valutazione) : void
+getVoto() : int
+setVoto(voto) : void
+getId_progetto() : int
+setId_progetto(id_progetto) : void
+getUsername() : String
+setUsername(username) : void
+toString() : String
ValcomModelDM
+getNumeroValutazioniByIdProgetto(idProgetto) : Integer
+getNumeroCommentiByIdProgetto(idProgetto) : Integer
+getNumeroRisposteByIdCommento(idCommento) : Integer
+getMediaValutazioniById(id) : Integer
+getCommentiByIdProgetto(idProgetto) : ArrayList<Commento>
+getRisposteByIdCommento(idCommento) : ArrayList<RispostaCommento>
+inserisciCommento(commento, idProgetto) : void
+getLastIdCommento() : int
+inserisciRisposta(risposta, idCommento) : void
+getLastIdRisposta() : int
+cancellaRisposta(idRisposta) : void
+cancellaCommento(idCommento) : void
+cancellaRisposteByIdCommento(idCommento) : void
+inserisciValutazione(valutazione, idProgetto) : void
+getLastIdValutazione() : int
+isValutato(progetto, utente) : boolean
+aggiornaValutazione(valutazione, idProgetto) : void
+eliminaValutazione(idProgetto, username) : void
+creaNotificaCommento(commento) : Notifica
+getLastIdNotifica() : int
+creaNotificaRisposta(risposta) : Notifica
+getCommentoById(idCommento) : Commento
+creaNotificaValutazione(valutazione) : Notifica
+getNotificheByUsername(username) : ArrayList<Notifica>
Notifica
-id_notifica : int
-immagine : Blob
-titolo : String
-tipo : String
-isClicked : int
-id_commento : int
-id_risposta : int
-id_progetto : int
-id_valutazione : int
-username : String
+getId_notifica() : int
+setId_notifica(id_notifica) : void
+getImmagine() : Blob
+setImmagine(immagine) : void
+getTitolo() : String
+setTitolo(titolo) : void
+getTipo() : String
+setTipo(tipo) : void
+getIsClicked() : int
+setIsClicked(isClicked) : void
+getId_commento() : int
+setId_commento(id_commento) : void
+getId_risposta() : int
+setId_risposta(id_risposta) : void
+getId_progetto() : int
+setId_progetto(id_progetto) : void
+getId_valutazione() : int
+setId_valutazione(id_valutazione) : void
+getUsername() : String
+setUsername(username) : void
+toString() : String
RispostaCommento
-id_risposta : int
-contenuto : String
-username : String
-id_commento : int
+getId_risposta() : int
+setId_risposta(id_risposta) : void
+getContenuto() : String
+setContenuto(contenuto) : void
+getUsername() : String
+setUsername(username) : void
+getId_commento() : int
+setId_commento(id_commento) : void
+toString() : String
Commento
-id_commento : int
-contenuto : String
-username : String
-id_progetto : int
+getId_commento() : int
+setId_commento(id_commento) : void
+getContenuto() : String
+setContenuto(contenuto) : void
+getUsername() : String
+setUsername(username) : void
+setId_progetto(id_progetto) : void
+getId_progetto() : int
+toString() : String

1.5 Definizioni, acronimi e abbreviazioni

- **HTML**: Linguaggio di mark-up per pagine web
- **JSP**: JavaServer Pages, tecnologia di programmazione Web in Java per lo sviluppo della logica di presentazione
- **COTS**: Component Off The Shelf
- **CSS**: Linguaggio usato per definire la formattazione di pagine web
- **Javascript**: Linguaggio di scripting orientato agli oggetti e agli eventi, comunemente utilizzato nella programmazione Web lato client per la creazione, in siti web e applicazioni web.
- **MD5**: Funzione hash crittografica.
- **Ecplise**: ambiente di sviluppo integrato multi-linguaggio e multi-piattaforma
- **Javadoc**: è un applicativo incluso nel Java Development Kit utilizzato per la generazione automatica della documentazione del codice sorgente scritto in linguaggio Java.
- **Off-The-Shelf**: Servizi esterni di cui viene fatto utilizzo da terzi.

1.6 Riferimenti

- Bernd Bruegge & Allen H. Dutoit, Object-Oriented Software Engineering: Using UML, Patterns and Java, (2nd edition), Prentice-Hall, 2003
- Ian Sommerville, Software Engineering, Addison Wesely
- SDD_Simplify3D

2. Packages

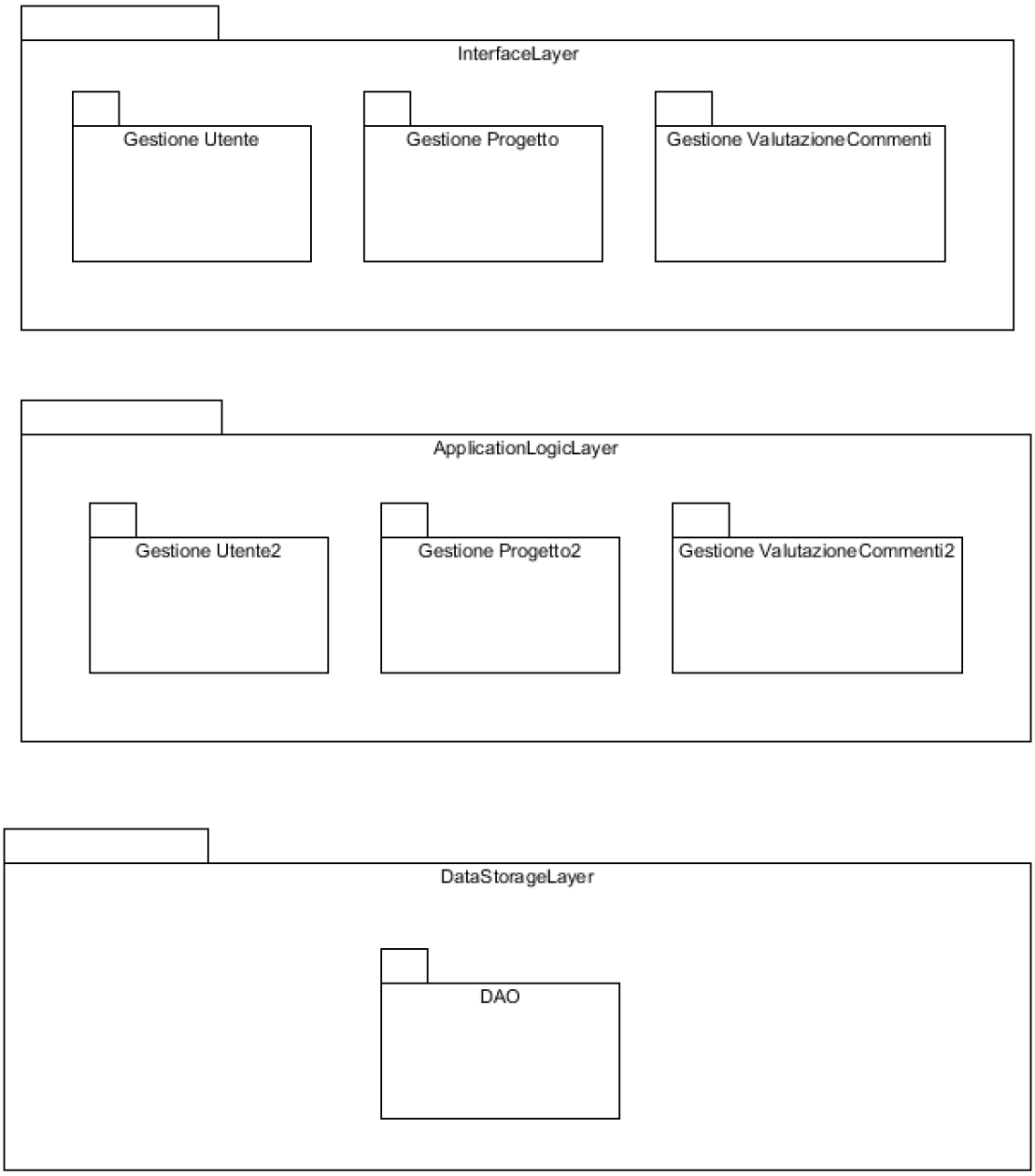
Per Simplify3D è stata adottata la seguente suddivisione in pacchetti, scomponendo principalmente il sistema secondo l'architettura three-layer adottata.

- **Interface Layer**: che contiene gli oggetti boundary, ovvero le view dei tre sottosistemi individuati (Gestione Utente, Gestione Progetto, Gestione Valutazione e Commento).

- **Application Logic Layer**: contiene la logica dei tre sottosistemi individuati (Gestione Utente, Gestione Progetto, Gestione Valutazione e Commento) con JavaBean e Servlet.

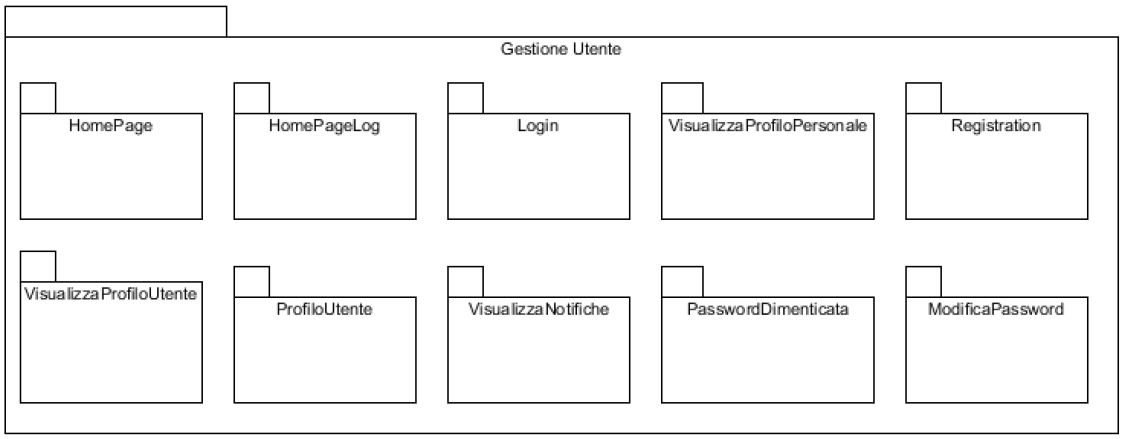
- **Data Storage Layer**: Formato dalle classi principali che si occuperanno di interfacciarsi al database e di passare le varie query.

2.1 General Package Diagram

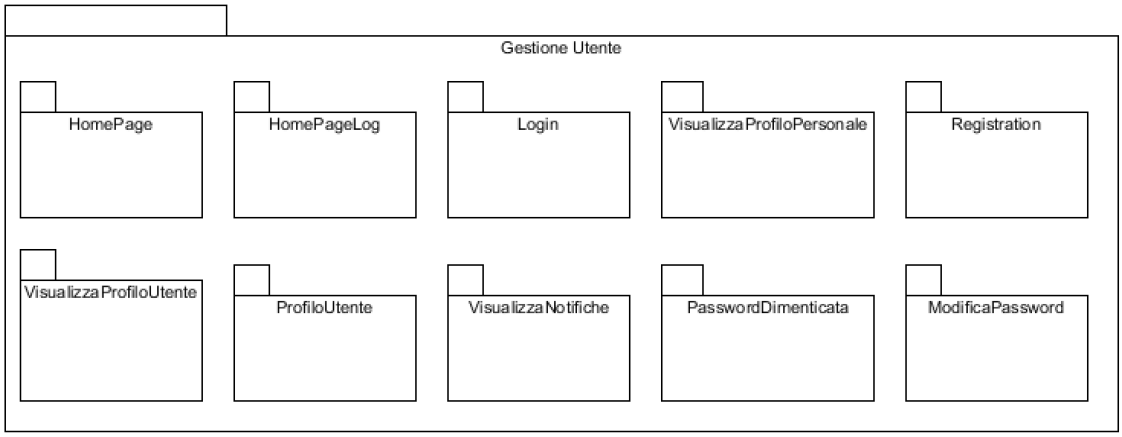


2.2 PK_InterfaceLayer

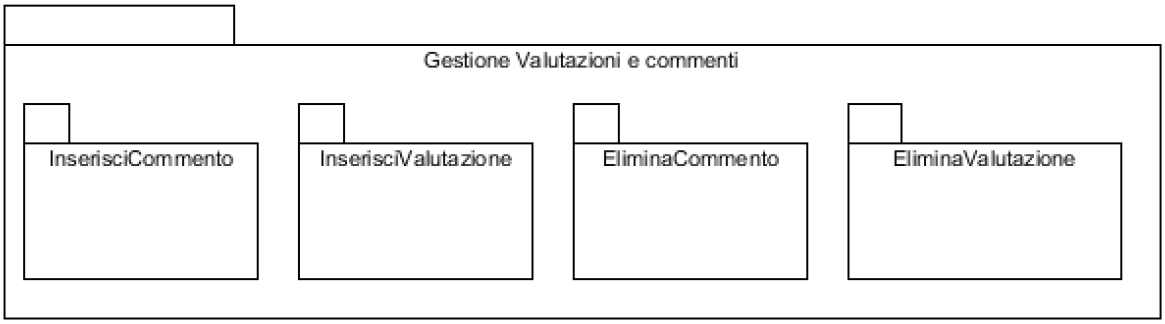
2.2.1 PK_Interface_GestioneUtente



2.2.2 PK_Interface_GestioneProgetto

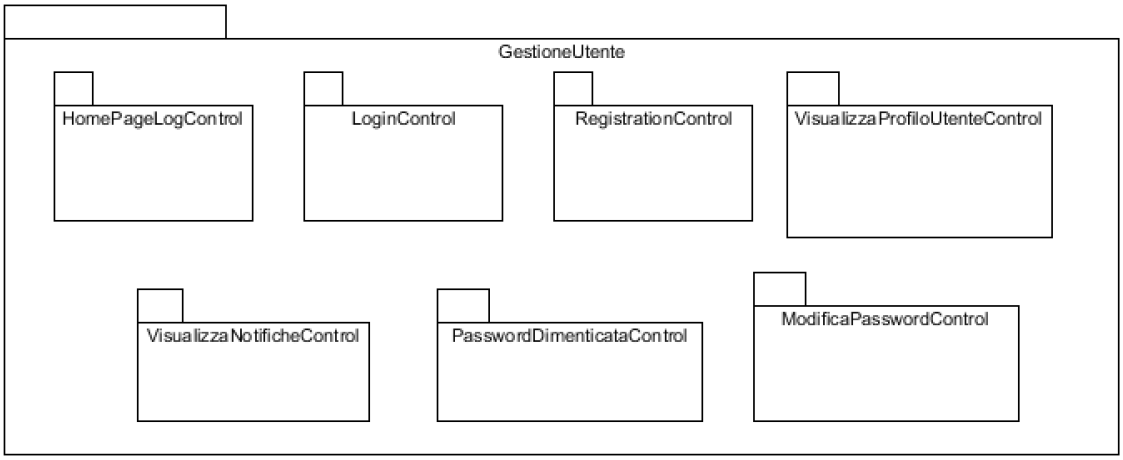


2.2.3 PK_Interface_GestioneValutazioneCommenti

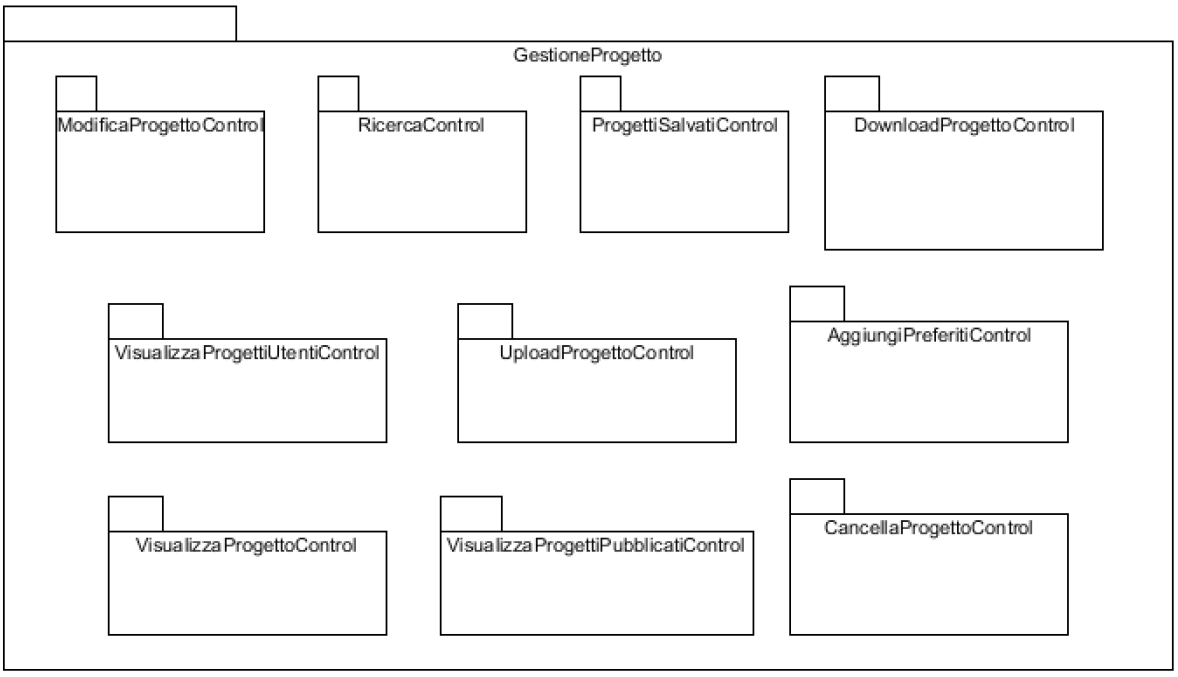


2.3 PK_ApplicationLogicLayer

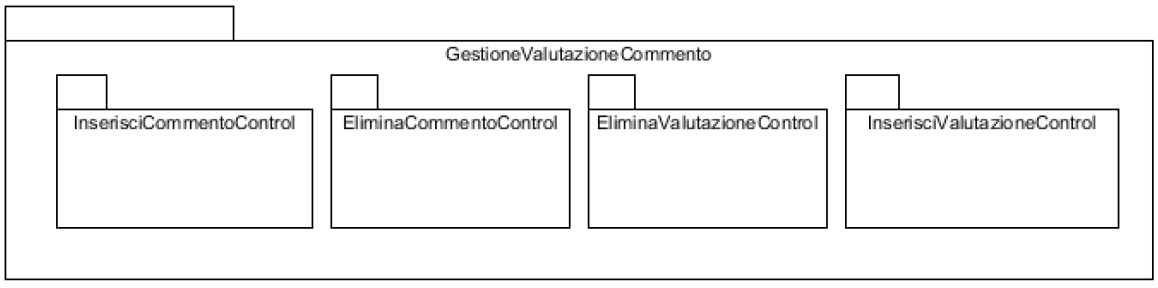
2.3.1 PK_Logic_GestioneUtente



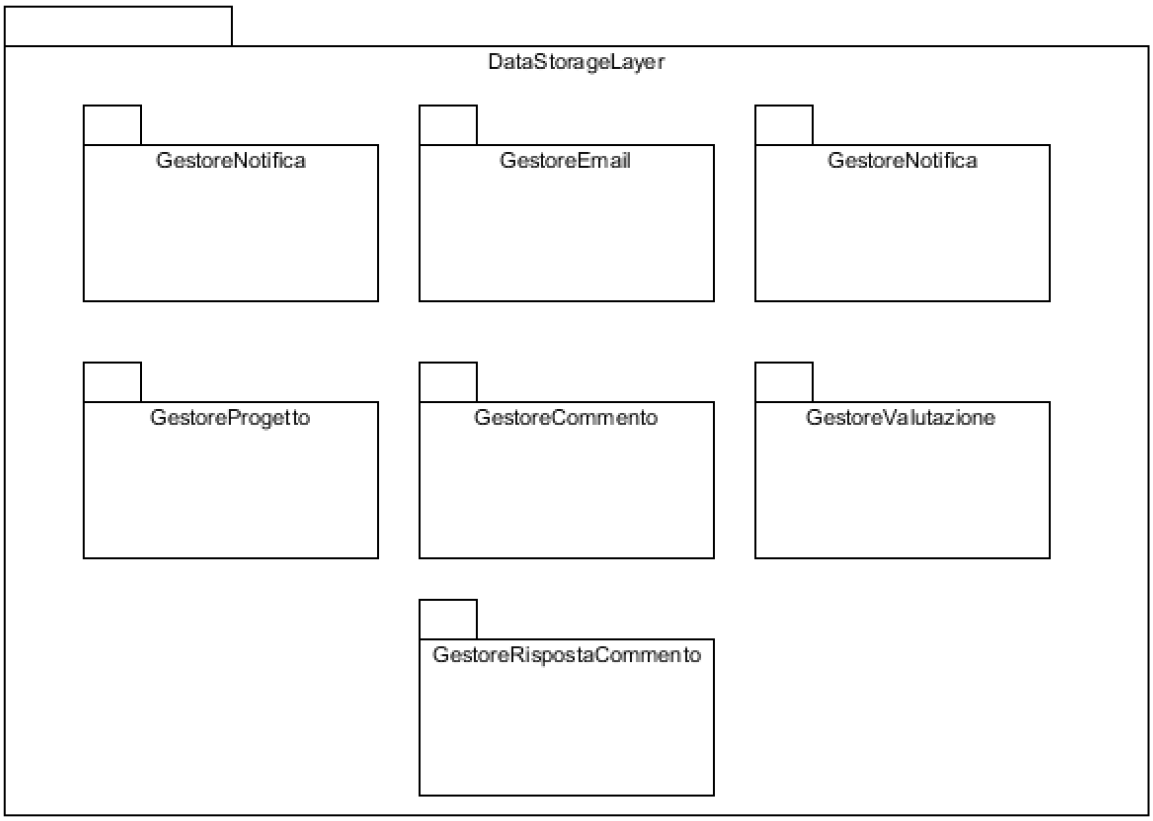
2.3.2 PK_Logic_GestioneProgetto



2.3.3 PK_Logic_GestioneValutazioneCommento



2.4 PK_StorageLayer



3. Interfacce delle classi

Nome classe	Utente
Descrizione	Questa classe rappresenta l'utente registrato al sito
Attributi	- username: String - cognome: String - nome: String - data_nascita: String - isAdmin: int - password: String - email: String - nazionalita: String - confermato: int - codice: int
Signature dei metodi	+ getUsername() + setUsername(username) + getCognome() + setCognome(String cognome) + getNome() + setNome(String nome) + getData_nascita() + setData_nascita(String data_nascita) + getIsAdmin() + setIsAdmin(int IsAdmin) + getPassword() + setPassword(String password) + getEmail() + setEmail(String email) + getNazionalita() + setNazionalita(String nazionalità) + getConfermato() + setConfermato(int confermato) + getCodice() + setCodice(int codifce)
Pre-condizioni	
Post-condizioni	
Invarianti	

Nome classe	Progetto
Descrizione	Questa classe rappresenta un progetto
Attributi	- id_progetto: int - titolo: String - descrizione: String - file_modello: Blob - immagine: Blob - consigli: String - categoria: String - versione: int - username: String
Signature dei metodi	+ getId_progetto() + setId_progetto(int id_progetto) + getTitolo() + setTitolo(String titolo) + getDescrizione() + setDescrizione(String descrizione) + getFile_modello() + setFileModello(Blob file_modello) + getImmagine() + setImmagine(Blob immagine) + getConsigli() + setConsigli(String consigli) + getCategoria() + setCategoria(String categoria) + getVersione() + setVersione(int versione) + getUsername() + setUsername(String username)
Pre-condizioni	
Post-condizioni	
Invarianti	

Nome classe	Valutazione
Descrizione	Questa classe rappresenta una valutazione
Attributi	- id_valutazione: int - voto: int - id_progetto: int - username: String
Signature dei metodi	+ getId_valutazione() + setId_valutazione(int id_valutazione) + getVoto() + setVoto(int voto) + getId_progetto() + setId_progetto(int id_progetto) + getUsername() + setUsername(String username)
Pre-condizioni	
Post-condizioni	
Invarianti	

Nome classe	Commento
Descrizione	Questa classe rappresenta un commento
Attributi	- id_commento: int - contenuto: String - username: String - id_progetto: int
Signature dei metodi	+ getId_commento() + setId_commento(int id_commento) + getContenuto() + setContenuto(String contenuto) + getUsername() + setUsername(String username) + getId_progetto() + setId_progetto(int id_progetto)
Pre-condizioni	
Post-condizioni	
Invarianti	

Nome classe	Risposta Commento
Descrizione	Questa classe rappresenta una risposta ad un commento
Attributi	- id_risposta: int - contenuto: String - username: String - id_commento: int
Signature dei metodi	+ getId_risposta() + setId_risposta(int id_risposta) + getContenuto() + setContenuto(String contenuto) + getUsername() + setUsername(String username) + getId_commento() + setId_commento(int id_commento)
Pre-condizioni	
Post-condizioni	
Invarianti	

Nome classe	Notifica
Descrizione	Questa classe riguarda una notifica
Attributi	- id_notifica: int - immagine: Blob - titolo: String - tipo: String - isClicked: int - id_commento: int - id_risposta: int - id_progetto: int - id_valutazione: int - username: String
Signature dei metodi	+ getId_notifica() + setId_notifica(int id_notifica) + getImmagine() + setImmagine(Blob immagine) + getTitolo() + setTitolo(String titolo) + getTipo() + setTipo(String tipo) + getTitolo() + setIsClicked (int isClicked) + getId_commento() + setId_commento(int id_commento) + getId_risposta()

	+ setId_risposta(int id_risposta) + getId_progetto() + setId_progetto(int id_progetto) + getId_valutazione() + setId_valutazione(int id_valutazione) + getUsername() + setUsername(String username)
Pre-condizioni	
Post-condizioni	
Invarianti	

Nome classe	UtenteModelDM
Descrizione	Questa classe gestisce le interazioni della classe Utente con il database
Attributi	- connection: Connection
Signature dei metodi	+ doRegistrazione(Utente utente, int codice) + doLogin(String username, String password) + doPasswordDimenticata(Utente utente, String nuovapassword) + doModificaPassword(Utente utente, String nuovapassword) + getUtenteByUsername(String username) + verificaCodice(String username) + setConfermato(Utente utente)
Pre-condizioni	**Context**: Utente: doRegistrazione(utente, codice); **pre**: utente non deve avere altre corrispondenze nel database **Context**: Utente: doLogin(username, password); **pre**: username e password devono avere una corrispondenza nel database e la registrazione deve essere confermata (confermato=1); **Context**: Utente: doPasswordDimenticata(username, email); **pre**: username e email devono avere una corrispondenza nel database;
Post-condizioni	
Invarianti	

Nome classe	ProgettoModelDM
Descrizione	Questa classe gestisce le interazioni della classe Progetto con il database
Attributi	- connection: Connection
Signature dei metodi	+ doUpload(Progetto progetto, InputStream file_modello, InputStream immagine) + getLastId( ) + getMostRated( ) + getByCategoria(String categoria) + getByUsername(String username) + getProgettoById(int id) + modificaProgetto(Progetto p) + addToPreferiti(Progetto progetto, Utente utente) + aggiornaDownload(Progetto progetto, Utente utente) + getDownloadById(int id) + removeFromPreferiti(Progetto progetto, Utente utente) + isPreferito(Progetto progetto, Utente utente) + getPreferitiByUsername(String username) + doCancellaProgetto(int id, String username) + ricercaBarra(String contenuto) + modificaProgetto(Progetto p, InputStream imagine, InputStream file_modello)
Pre-condizioni	**Context**: Progetto: getByCategoria(categoria); **pre**: devono esserci corrispondenze tra i progetti e categoria nel database; **Context**: Progetto: getByUsername(username); **pre**: devono esserci corrispondenze tra i progetti e username nel database; **Context**: Progetto: getProgettoById(id); **pre**: deve esserci una corrispondenza tra i progetti e l'id nel database; **Context**: Progetto: removeFromPreferiti(progetto, utente); **pre**: deve esserci una corrispondenza tra il progetto e l'utente nella tabella preferiti del database;
Post-condizioni	
Invarianti	

Nome classe	ValcomModelDM
Descrizione	Questa classe gestisce le interazioni delle classi Valutazione, Commenti, Notifica, RispostaCommento con il database
Attributi	- connection: Connection
Signature dei metodi	+ getNumeroValutazioniByIdProgetto(int idProgetto) + getNumeroCommentiByIdProgetto(int idProgetto) + getNumeroRisposteByIdCommento(int idCommento) + getMediaValutazioniById(int id) + getCommentiByIdProgetto(int idProgetto) + getRisposteByIdCommento(int idCommento) + inserisciCommento(Commento comment, int idProgetto) + getLastIdCommento( ) + inserisciRisposta(RispostaCommento risposta, int idCommento) + getLastIdRisposta( ) + cancellaRisposta(int idRisposta) + cancellaCommento(int idCommento) + cancellaRispostaByIdCommento(int idCommento) + inserisciValutazione(Valutazione valutazione, int idProgetto) + getLastIdValutazione ( ) + isValutato(Progetto progetto, Utente utente) + aggiornaValutazione(Valutazione valutazione, int idProgetto) + eliminaValutazione(int idProgetto, String username) + creaNotificaCommento(Commento commento) + getLastIdNotifica( ) + creaNotificaRisposta(RispostaCommento risposta) + getCommentoById(int idCommento) + creaNotificaValutazione(Valutazione valutazione) + getNotificheByUsername(String username) + getNumeroNotificheNonLette(String username) + setClickedNotifica(Notifica notifica)
Pre-condizioni	**Context**: RispostaCommento: inserisciRisposta(risposta, idCommento); **pre**: deve essere presente il commento nel database per inserire la risposta; **Context**: Valutazione: aggiornaValutazione(valutazione, idProgetto); **pre**: deve essere presente una valutazione al progetto nel database per poterla aggiornare; **Context**: Valutazione: inserisciValutazione(valutazione, idProgetto); **pre**: non deve essere presente una valutazione al progetto nel database per poterla inserire;
Post-condizioni	
Invarianti	