Laurea in informatica-Università di Salerno
Corso di *Ingegneria del Software*- Prof.ssa F.Ferrucci

DocuExchange

ODD: Object Design Document

DocuExchange Unisa Informatica

Presentato da:

Emanuele Bruno
Giacomo Impronta
Michele de Rosa
Paolo Erra

Revision History

Data	Versione	Cambiamenti	Autori
11/12/19	0.1	Strutturazione documento	Tutti
11/12/19	0.2	Stesura capitolo 1 da paragafo 1.1 a 1.4.1	Paolo Erra
12/12/19	0.3	Stesura da paragrafo 1.4.2 a 1.5	Giacomo Impronta, Paolo Erra
12/12/19	0.4	Stesura capitolo 2 e capitolo 3	Paolo Erra, Michele de Rosa, Giacomo Impronta
12/12/19	0.5	Stesura Class Diagram e glossario	Paolo Erra, Michele de Rosa, Giacomo Impronta
08/01/2020	0.6	Correzione Schemi e revisione documento	Giacomo Impronta
23/01/2020	0.7	Correzione Paragrafo “controller” e revisione schemi documento	Giacomo Impronta

Sommario

1. Introduzione

1.1 Object Design trade-off

Nella fase dell'Object Design verranno fatte importanti scelte di progettazione, è necessario determinare quali punti rispettare e quali rendere opzionali.
Per la realizzazione sono stati riconosciuti i seguenti trade-off:

Memoria vs Estensibilità
Il sistema sarà implementato in modo tale da preferire l'estensibilità alla memoria così da agevolare lo sviluppo di funzionalità addizionali dando meno importanza alla memoria utilizzata da quest'ultime.

Tempo di risposta vs Affidabilità
Il sistema sarà implementato preferendo l'affidabilità al tempo di risposta in quanto vi saranno controlli più accurati dei dati in input a discapito del tempo di risposta.

Interfaccia vs Easy-use
La piattaforma è molto semplice e di facile utilizzo poiché ha un'interfaccia chiara e intuitiva.

Disponibilità vs Tolleranza ai guasti:
Il sistema sarà implementato in modo tale da preferire la disponibilità alla tolleranza ai guasti in quanto il sistema dovrà essere sempre disponibile all'utente anche in caso di errore durante l'uso di una funzionalità, anche al costo di rendere non disponibile quest'ultima per un lasso di tempo.

Criteri di manutenzione vs Criteri di performance
Il sistema sarà implementato in modo tale da preferire la manutenibilità alla performance in quanto il nostro obiettivo è quello di facilitare gli sviluppatori nel processo di aggiornamento del software a discapito delle performance del sistema.

1.2 Componenti off-the-shelf

Per il sistema che si vuole realizzare faremo uso di componenti **off-the-shelf**, ovvero componenti software già disponibili per facilitare l'implementazione del software.
Per la parte grafica utilizzeremo Bootstrap, un framework open source che contiene una raccolta di strumenti di libero utilizzo per la creazione di siti e applicazione per il Web. Questo framework contiene tools basati su HTML e CSS, sia per la tipografia che per le componenti dell'interfaccia come per esempio moduli, bottoni per la navigazione, così come alcune estensioni opzionali di JavaScript.
Per ottenere alcuni effetti visuali nonché velocizzare il sistema utilizzeremo inoltre JQuery e JavaScript che permetteranno all'interfaccia di rispondere alle azioni dell'utente agilmente, e un miglioramento dell'esperienza con le tecniche basate su Ajax.
Inoltre, per gestire le connessioni, verrà utilizzato un sistema di connection pool, fornito da tomcat, riferimenti: http://tomcat.apache.org/tomcat-9.0-doc/jdbc-pool.html. Risulta senza alcun'ombra di dubbio, più efficiente e veloce utilizzare una componente preesistente e fornita da un ente riconosciuto.

1.3 Interface Documentation guidelines

Per l'implementazione del sistema, gli sviluppatori dovranno rispettare le seguenti linee guida

1.3.1 Classi e interfacce Java

Per quanto riguarda la scrittura del codice per le classi Java si rispetterà lo standard Google Java (http://google.github.io/styleguide/javaguide.html#s4.6-whitespace). Inoltre ci sarà la possibilità di esserci commenti in merito a specifiche decisioni o calcoli.
Per la scrittura dei nomi delle variabili verrà utilizzata la nota convenzione lowerCamelCase, tale convenzione consiste nello scrivere parole composte o frasi unendo tutte le parole tra loro.

Per le implementazioni delle classi e interfacce Java, si devono rispettare le seguenti regole di formattazione:

- La parentesi graffa aperta "{" si trova alla fine della stessa linea dell'istruzione di dichiarazione.
- La parentesi graffa chiusa "}" si trova all'inizio di una nuova riga vuota nello stesso livello di indentazione del nome della classe o interfaccia.
- Non devono essere inseriti spazi tra il nome del metodo e la parentesi tonda che apre la lista dei parametri.

Nel caso di istruzioni semplici:

- Ogni linea deve contenere una sola istruzione.

Nel caso di istruzioni composte:

- Bisogna indentare correttamente le istruzioni racchiuse all'interno di un blocco.
- Le parentesi di apertura del blocco deve trovarsi alla fine della riga dell'istruzione composta.
- La parentesi di chiusura del blocco deve trovarsi all'inizio di una nuova riga dopo il blocco di istruzioni.

I nomi delle classi e dei metodi devono essere semplici e descrittivi in modo tale da descrivere al meglio il dominio applicativo. I nomi dei metodi iniziano con una lettera minuscola e seguono la notazione a cammello. Non saranno consentiti caratteri speciali.

```
return()-> {
        while(condition()){
                method();
        }
};

return new MyClass() {
        @Override public void method() {
                if(condition()) {
                        something();
                }
                else if(otherCondition){
                        somethingElse();
                }
        }
};
```

1.3.2 Indentazione

Qualunque sia il linguaggio usato per la produzione di codice, ogni istruzione deve essere opportunamente indentata. L'indentazione deve essere effettuata con un TAB.

Esempio:

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

Deve essere sostituita da:

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

In particolare, l'indentazione deve essere molto accurata per le istruzioni FOR, WHILE e IF per garantire una buona leggibilità del codice.

1.3.3 Parentesi

Per quanto riguarda le parentesi nella scrittura di classi java, nei costrutti FOR, WHILE e IF, è necessario, anche se ci fosse una sola istruzione, riportare il blocco di istruzioni tra parentesi graffe per garantire una buona leggibilità del codice.

1.3.4 Pagine lato Server (JSP)

Le pagine JSP devono essere conforme allo standard HTML 5. Per quanto riguarda il codice Java presente nelle pagine JSP deve rispettare le seguenti regole:

- Il tag di apertura (<%) è seguito immediatamente dalla fine della riga;
- Il tag di chiusura (%>) si trova all'inizio di una riga;
- È possibile evitare le due regole precedenti, se il corpo del codice Java consiste in una singola istruzione (<%= %>):

Per le Servlet è necessario far terminare il nome della classe con la parola "Servlet".

1.3.5 Fogli di stile CSS

I fogli di stile (CSS) devono seguire le seguenti regole:

- I selettori della regola si trovano a livello 0 di indentazione, uno per riga;
- Le proprietà che costituiscono la regola sono listate una per riga e sono indentate rispetto ai selettori;
- La regola è terminata da una parentesi graffa chiusa (}), collocata da sola su una riga;

1.3.6 Database SQL

Si fornisce un insieme di regole per quanto riguarda la realizzazione e la gestione delle tabelle del database.

Per le tabelle del database vanno rispettate le seguenti regole:

- I nomi delle tabelle devono essere formati esclusivamente da lettere maiuscole;
- Il nome della tabella deve essere un sostantivo singolare e deve descrivere al meglio l'entità del dominio applicativo;

Per i nomi dei campi banno rispettate le seguenti regole:

- I nomi dei campi dei campi delle tabelle devono essere formati esclusivamente da lettere maiuscole;
- Se il nome del campo di una tabella è costituito da più parole, è previsto l'uso di underscore (_);
- Il nome del campo deve essere un sostantivo singolare tratto dal dominio del problema ed esplicativo del contenuto;

1.4 Design Pattern

1.4.1 MVC

Il design pattern MVC consente la suddivisione del sistema in tre blocchi principali: Model, View e Controller.

- **Model:** Ilpackage model rappresenta i dati e le regole di business che gestiscono l'interazione con i dati stessi ed espone al controller e alla view le funzioni per accedervi. L'unico responsabile dell'accesso ai dati è il model per cui ogni richiesta di accesso o di manipolazione dei dati stessi deve essere eseguita attraverso di esso.

- **View:** si occupa dell'interazione con l'utente e della presentazione dei dati prelevati dal Model, visualizzando all'utente gli oggetti del dominio applicativo

- **Controller:** riceve i comandi dell'utente attraverso il View e modifica lo stato di quest'ultimo e del Model. Nel nostro sistema le classi sono state divise in package avente nome richiamante il blocco di appartenenza nel design pattern.

1.5 Definizioni, Acronimi e abbreviazioni

JSP: acronimo di Java Scripting Preprocessor, è una tecnologia di programmazione web in java per lo sviluppo della logica di presentazione (tipicamente secondo il pattern MVC) di applicazioni web.

MVC: acronimo di Model-view-controller, è un pattern architetturale molto diffuso nello sviluppo di sistemi software**.**

Off-The-Shelf: Servizi esterni al sistema di cui viene fatto utilizzo.

Bootstrap: è una raccolta di strumenti liberi per la creazione di siti e applicazioni per il web.

HTML: Linguaggio di programmazione utilizzato per lo sviluppo di pagine Web.

CSS: acronimo di Cascading Style Sheets è un linguaggio usato per definire la formattazione delle pagine Web.

JavaScript: JavaScript è un linguaggio di scripting orientato agli oggetti e agli eventi, comunemente utilizzato nella programmazione Web lato client per la creazione di effetti dinamici interattivi.

JQuery: JQuery è una libreria JavaScript per applicazioni web.
AJAX: AJAX, acronimo di Asynchronous JavaScript and XML, è una tecnica di sviluppo software per la realizzazione di applicazioni web interattive.

lowerCamelCase: Il lowerCamelCase è una tecnica di naming delle variabili adottata dallo standard Google Java. Essa consiste nello scrivere più parole insieme delimitando la fine e l'inizio di una nuova parola con una lettera maiuscola.

Servlet: le Servlet sono oggetti scritti in linguaggio Java che operano all'interno di un server web.

Tomcat: Apache Tomcat è un web server open source. Implementa le specifiche JavaServer Pages (JSP) e servlet, fornendo quindi una piattaforma software per l'esecuzione di applicazioni Web sviluppate in linguaggio Java.

1.7 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_DDI_1.0
- http://getbootstrap.com/
- http://www.highcharts.com/
- https://github.com/SheetJS/js-xlsx

2. Package

2.1 Model

Questo package conterrà a sua volta altri tre package:

- **Bean**
- **Interface**
- **DAO**

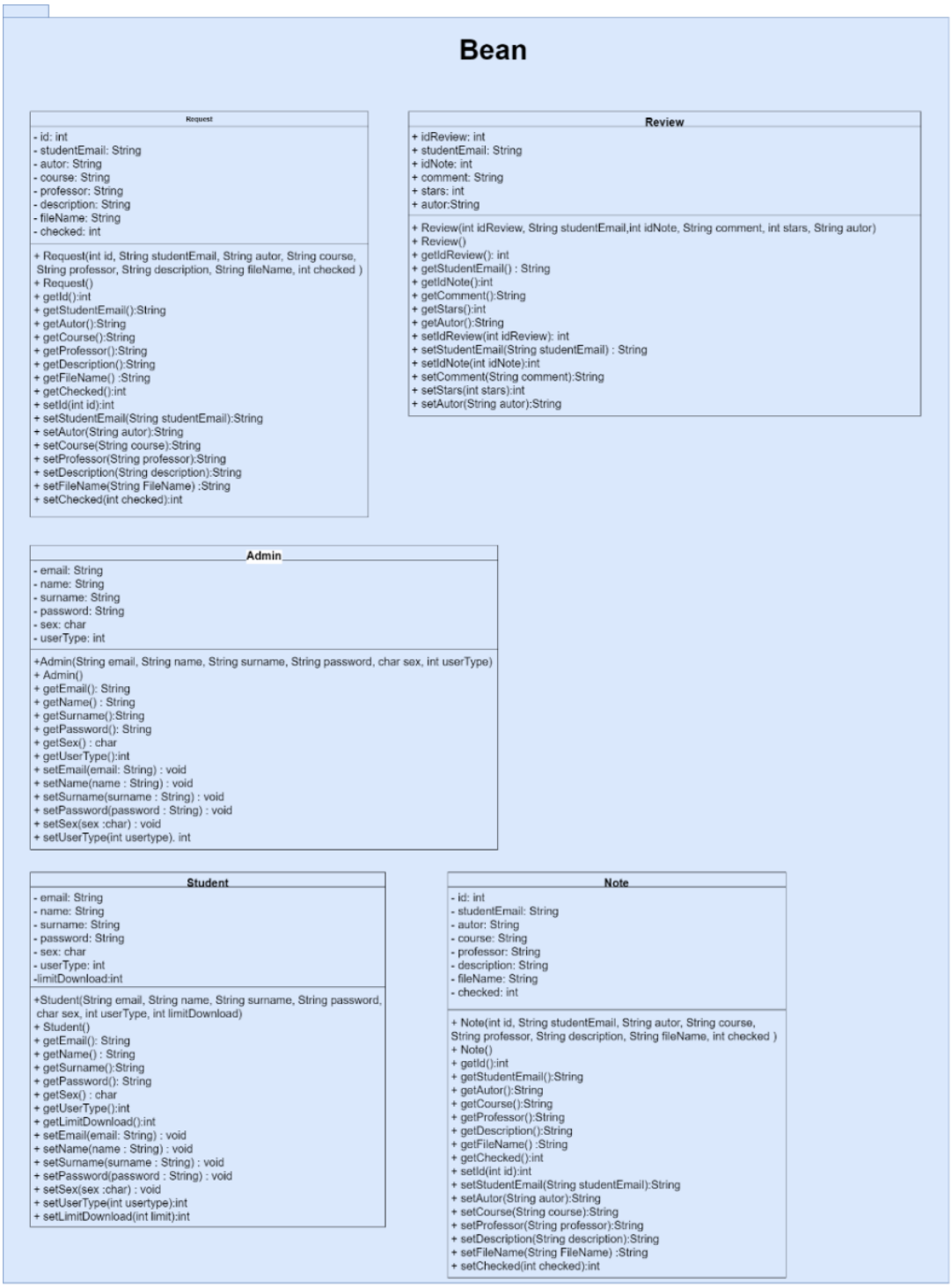
2.1.1 Package Bean

In questo package sono presenti tutti gli oggetti entity che definiscono gli oggetti di dominio e i quali saranno usati come bean dalle servlet presenti nel package controller.

- **Request:** Classe che rappresenta le informazioni relative alla richiesta, implementa l'interfaccia NoteInterface presente nel package Interface;
- **Note:** Classe che rappresenta le informazioni relativi alla richiesta e all'appunto, implementa l'interfaccia NoteInterface presente nel package Interface;
- **Review:** Classe che rappresenta le informazioni relativi alla recensione di un appunto, implementa

l'interfaccia ReviewInterface presente nel package Interface;

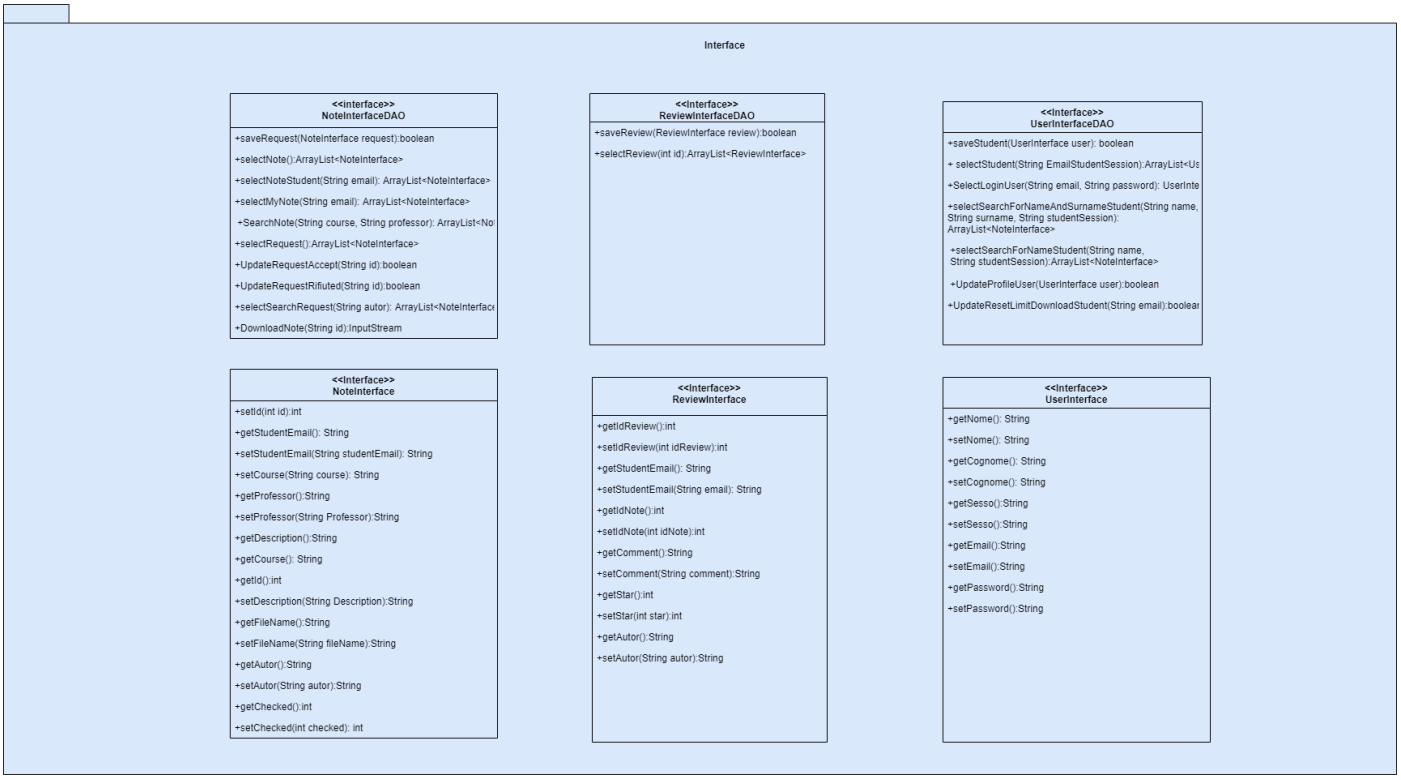
- **Student:** Classe che rappresenta le informazioni relativi allo studente, implementa l'interfaccia UserInterface presente nel package Interface;
- **Admin:** Classe che rappresenta le informazioni relativi all'admin, implementa l'interfaccia UserInterface presente nel package Interface;



2.1.2 Interface

Questo package contiene le interfacce che definiscono tutte le operazioni effettuabili sulle entità bean e dalle classi dao responsabili delle operazioni sul database.

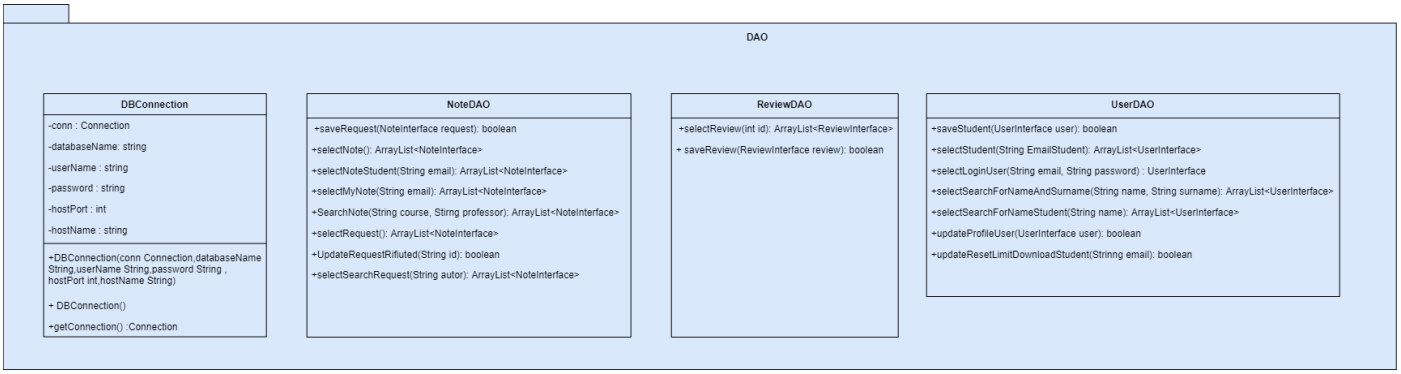
- **NoteInterface:** Interfaccia che definisce le operazioni effettuabili sulle entità Note e Request;
- **ReviewInterface:** Interfaccia che definisce le operazioni effettuabili sulle entità Review;
- **UserInterface:** Interfaccia che definisce le operazioni effettuabili sulle entità Admin e Student;
- **NoteInterfaceDAO:** Interfaccia che definisce le operazioni effettuabili per la classe NoteDAO;
- **ReviewInterfaceDAO:** Interfaccia che definisce le operazioni effettuabili per la classe ReviewDAO;
- **UserInterfaceDAO:** Interfaccia che definisce le operazioni effettuabili per la classe UserDAO;



2.1.3 Package DAO

Questo package contiene tutte le classi che implementano le interfacce DAO definendo le operazioni di gestione dei dati rappresentati dai bean.

- **NoteDAO:** Responsabile delle operazioni sul database relativi agli appunti e le richieste, implementa l'interfaccia NoteInterfaceDAO presente nel package Interface.
- **ReviewDAO:** Responsabile delle operazioni sul database relativi alle recensioni, implementa l'interfaccia ReviewInterfaceDAO presente nel package Interface.
- **UserDAO:** Responsabile delle operazioni sul database relativi agli utenti, implementa l'interfaccia UserInterfaceDAO presente nel package Interface.
- **DBConnection:** Permette alle altre classi del package di ricevere connessioni dal database.



2.2 View

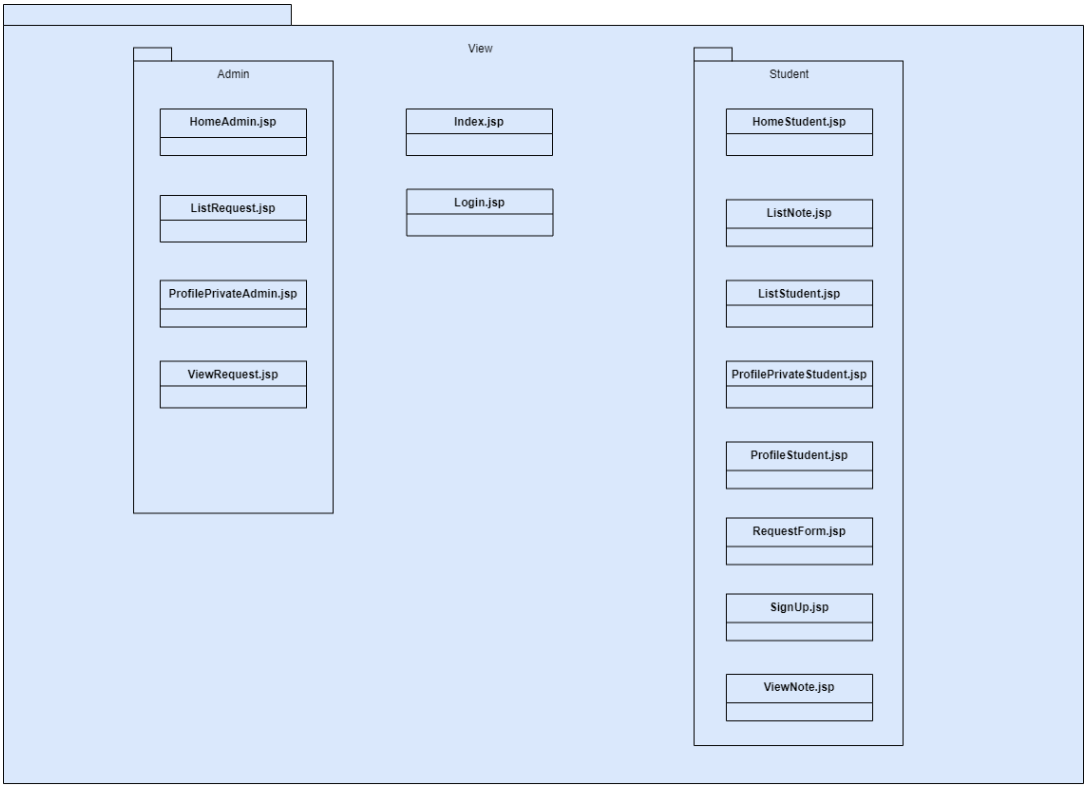
In questo sottosistema sono presenti le pagine JSP che serviranno per la visualizzazione della piattaforma.
Le pagine JSP sono categorizzate per il tipo di utente.

Pagine JSP Admin:

- **ViewRequest JSP:** permette di visualizzare una richiesta appunti in modo specifico;
- **HomeAdmin JSP:** permette di visualizzare la home page lato Admin;
- **ListRequest JSP:** permette di visualizzare tutte le richieste di pubblicazioni appunti;
- **ProfilePrivateAdmin JSP:** permette di visualizzare i dati dell'admin;

Pagine JSP Studente

- **HomeStudent JSP:** permette di visualizzare la home page lato Student;
- **ListNote JSP:** permette di visualizzare tutte le note pubblicate sulla piattaforma;
- **ProfilePrivateStudent JSP:** permette di visualizzare ed eventualmente modificare il proprio profilo utente;
- **ProfileStudent JSP:** permette di visualizzare il profilo di altri studenti;
- **RequestForm JSP:** permette di inoltrare una richiesta di condivisione appunti agli admin;
- **SignUp JSP**: permette all'utente di registrarsi alla piattaforma;
- **ViewNoteJSP:** permette di visualizzare ed eventualmente scaricare una nota in particolare;

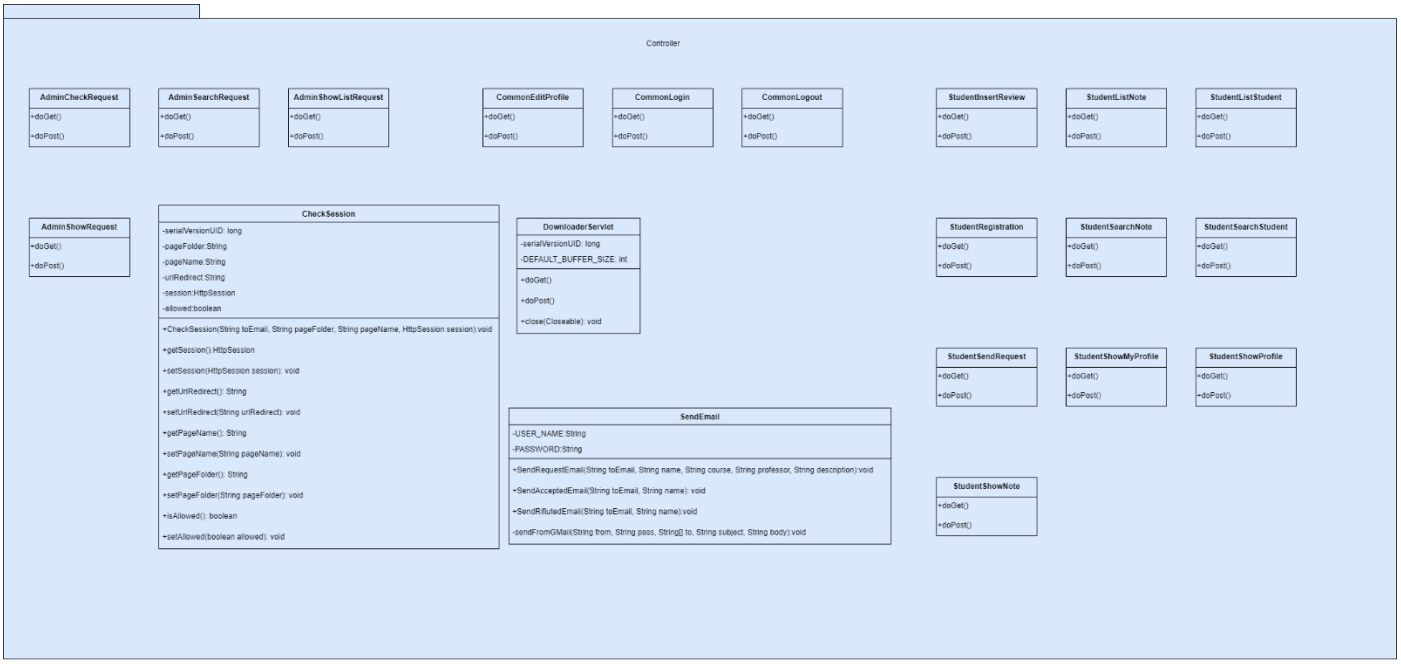


2.3 Controller

Il package controller riceve, contiene le servlet:

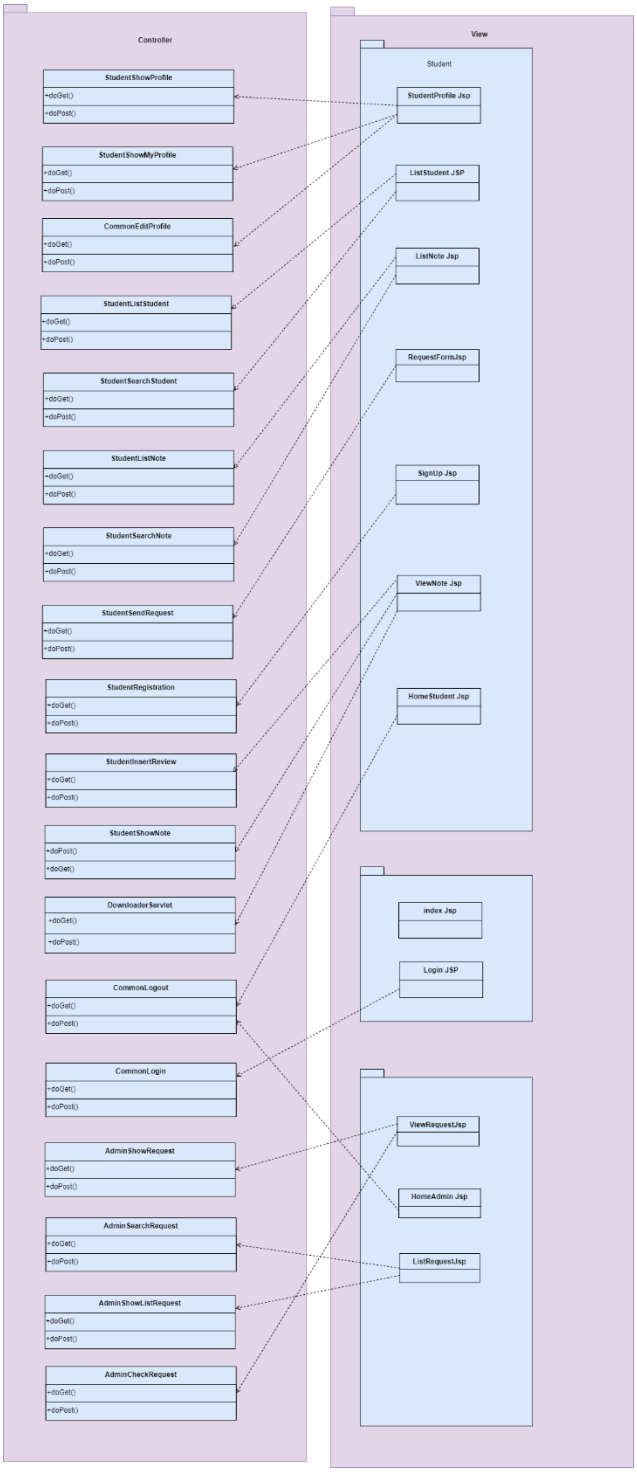
Nome	Descrizione
AdminCheckRequest	Permette all'admin di verificare una richiesta di pubblicazione da parte di uno studente. Se la richiesta è approvata viene pubblicato l'appunto, altrimenti è cancellato.
AdminSearchRequest:	Permette all'admin di ricercare una richiesta attraverso il nome dell'autore.
AdminShowListRequest:	Permette all'admin di visualizzare tutte le richieste di pubblicazione appunti.
AdminShowRequest	Permette all'admin di visualizzare una singola richiesta di condivisione appunti.
CheckSession	Serve per gestire le sessioni dell'utente evitando l'accesso a pagine non consentite.
CommonEditProfile	Permette a qualsiasi utente di modificare i propri dati.

CommonLogin	Permette a qualsiasi utente di effettuare il login.
CommonLogout	Permette a qualsiasi utente di effettuare il logout.
DownloaderServlet	Permette a tutti gli utenti di scaricare gli appunti.
SendEmail	Permette di inviare una mail agli studenti che hanno inoltrato una richiesta di condivisione appunti. Permette di inviare inoltre l'esito di una richiesta (dopo la visualizzazione da parte dell'admin) da parte di uno studente.
StudentInsertReview	Permette allo studente di scrivere una recensione.
StudentListNote	Permette allo studente di visualizzare la lista di tutti gli appunti pubblicati.
StudentListStudent	Permette allo studente di visualizzare la lista di tutti gli studenti iscritti alla piattaforma.
StudentRegistration	Permette all'utente di registrarsi come studente.
StudentSearchNote	Permette allo studente di ricercare gli appunti inserendo il nome del corso e/o il nome del professore.
StudentSearchStudent	Permette allo studente di ricercare altri studenti attraverso il nome e/o il cognome.
StudentSendRequest	Permette allo studente di inviare una richiesta di condivisione appunti.
StudentShowMyProfile	Permette allo studente di visualizzare il proprio profilo utente.
StudentShowNote	Permette allo studente di visualizzare un singolo appunto.
StudentShowProfile	Permette allo studente di visualizzare il profilo di un altro studente.

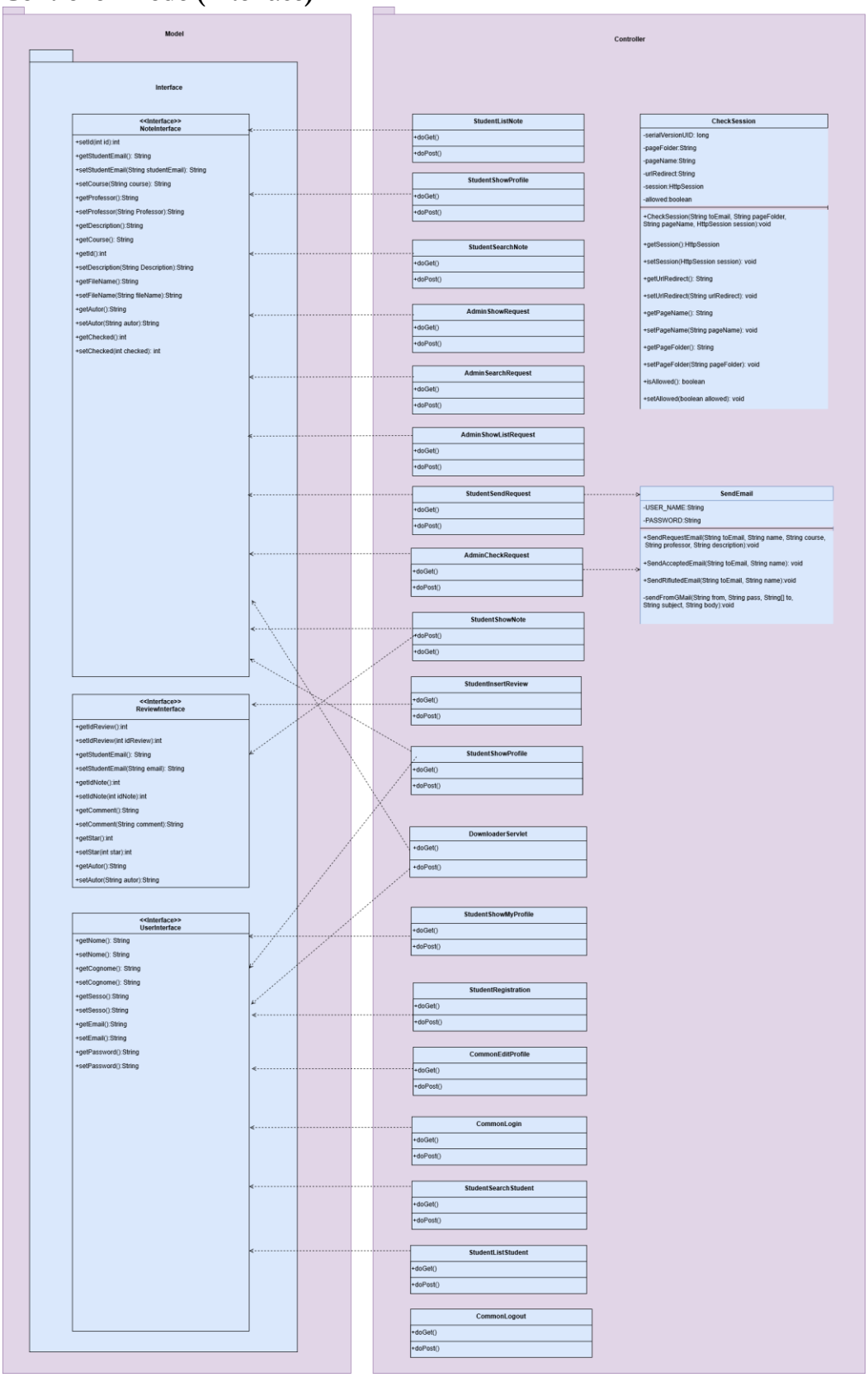

Controller
AdminCheckRequest
+doGet()
+doPost()
AdminSearchRequest
+doGet()
+doPost()
AdminShowListRequest
+doGet()
+doPost()
CommonEditProfile
+doGet()
+doPost()
CommonLogin
+doGet()
+doPost()
CommonLogout
+doGet()
+doPost()
StudentInsertReview
+doGet()
+doPost()
StudentListNote
+doGet()
+doPost()
StudentListStudent
+doGet()
+doPost()
AdminShowRequest
+doGet()
+doPost()
CheckSession
-serialVersionUID: long
-pageFolder:String
-pageName:String
-urlRedirect:String
-session:HttpSession
-allowed:boolean
+CheckSession(String toEmail, String pageFolder, String pageName, HttpSession session):void
+getSession():HttpSession
+setSession(HttpSession session): void
+getUrlRedirect(): String
+setUrlRedirect(String urlRedirect): void
+getPageName(): String
+setPageName(String pageName): void
+getPageFolder(): String
+setPageFolder(String pageFolder): void
+isAllowed(): boolean
+setAllowed(boolean allowed): void
DownloaderServlet
-serialVersionUID: long
-DEFAULT_BUFFER_SIZE: int
+doGet()
+doPost()
+close(Closeable): void
SendEmail
-USER_NAME:String
-PASSWORD:String
+SendRequestEmail(String toEmail, String name, String course, String professor, String description):void
+SendAcceptedEmail(String toEmail, String name): void
+SendRifiutedEmail(String toEmail, String name):void
-sendFromGMail(String from, String pass, String[] to, String subject, String body):void
StudentRegistration
+doGet()
+doPost()
StudentSearchNote
+doGet()
+doPost()
StudentSearchStudent
+doGet()
+doPost()
StudentSendRequest
+doGet()
+doPost()
StudentShowMyProfile
+doGet()
+doPost()
StudentShowProfile
+doGet()
+doPost()
StudentShowNote
+doGet()
+doPost()

2 Class diagram

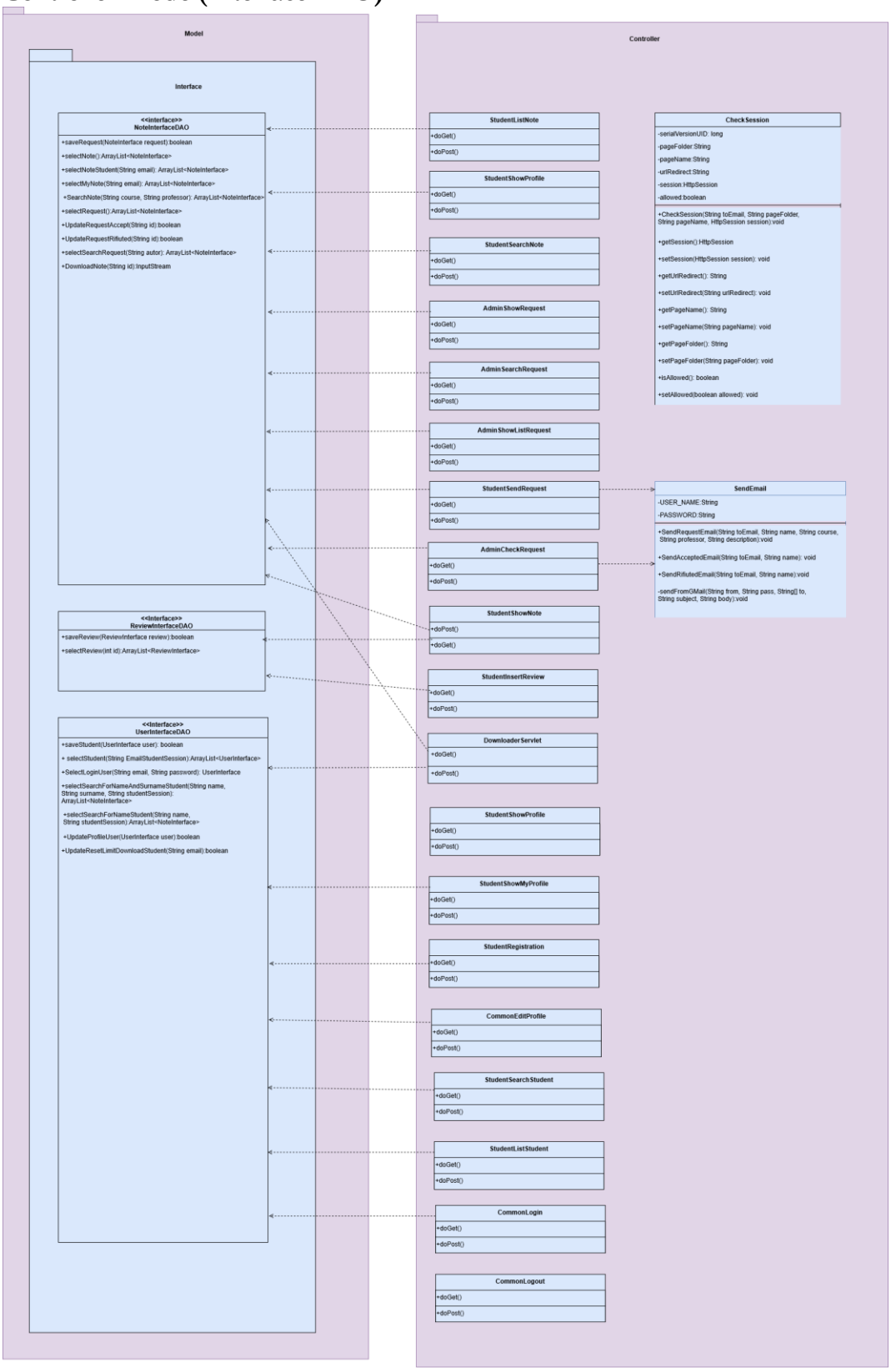
Controller-View



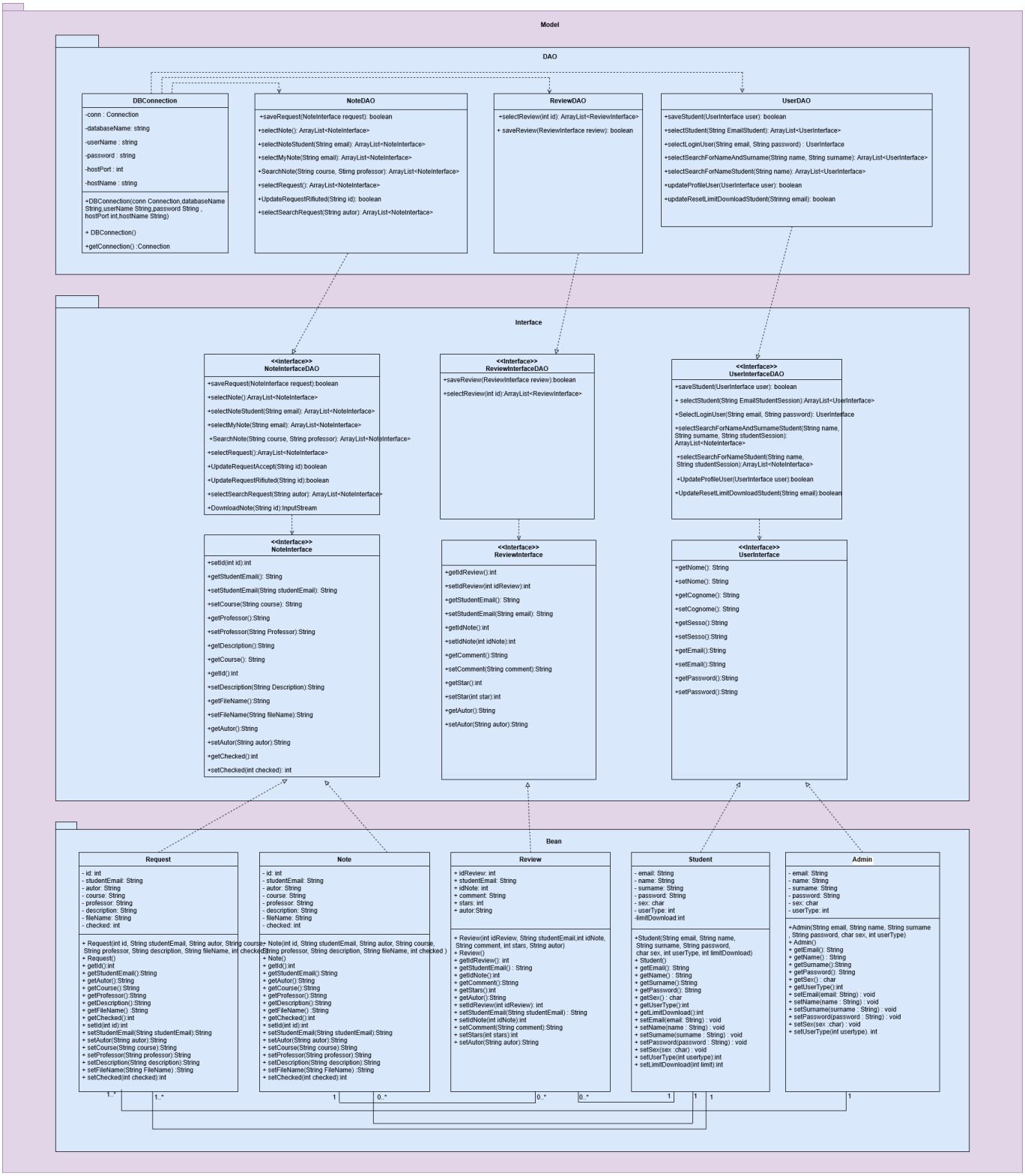
Controller-Model(Interface)



Controller-Model(Interface DAO)



Model Completo



3 Glossario

Trade-off: Il Trade-off è una situazione che implica una scelta tra due possibilità, in cui la perdita di valore di una costituisce un aumento di valore in un'altra.

Off-The-Shelf: Servizi esterni al sistema di cui viene fatto utilizzo.

Bootstrap: Framework che contiene librerie utili per lo sviluppo responsive di pagine web.

HTML: Linguaggio di programmazione utilizzato per lo sviluppo di pagine Web.

CSS: Linguaggio usato per definire la formattazione delle pagine Web.

JavaScript: JavaScript è un linguaggio di scripting orientato agli oggetti e agli eventi, comunemente utilizzato nella programmazione Web lato client per la creazione di effetti dinamici interattivi.

JQuery: JQuery è una libreria JavaScript per applicazioni web.

AJAX: AJAX, acronimo di Asynchronous JavaScript and XML, è una tecnica di sviluppo software per la realizzazione di applicazioni web interattive.

lowerCamelCase: Il lowerCamelCase è una tecnica di naming delle variabili adottata dallo standard Google Java. Essa consiste nello scrivere più parole insieme delimitando la fine e l'inizio di una nuova parola con una lettera maiuscola.

Servlet: i servlet sono oggetti scritti in linguaggio Java che operano all'interno di un server web.

Tomcat: Apache Tomcat è un web server open source. Implementa le specifiche JavaServer Pages (JSP) e servlet, fornendo quindi una piattaforma software per l'esecuzione di applicazioni Web sviluppate in linguaggio Java.