TutoratoSmart

ODD Object Design Document

TutoratoSmart

Riferimento	
Versione	0.5
Data	31/01/2020
Destinatario	Prof.ssa F. Ferrucci
Presentato da	Marco Delle Cave, Francesco Pagano, Manuel Pisciotta, Alessia Olivieri
Approvato da	

Data	Versione	Descrizione	Autori
11/12/19	0.1	Definizione ed introduzione ODD, scelta dei trade-off	[tutti]
14/12/19	0.2	Scelta design pattern, definizione interfacce delle classi e package	[tutti]
15/12/19	0.3	Revisione generale ODD	[tutti]
17/12/19	0.4	Rettifica ODD	[tutti]
31/01/20	0.5	Rettifica ODD	Pisciotta Manuel

Sommario

1. Introduzione

L'Object Design Document consente di specificare in modo dettagliato le decisioni prese in fase di analisi e di design; in particolare verranno specificati i principali trade-offs, descritte le componenti off-the-shelfs utilizzate dal sistema, le linee guida per la documentazione delle interfacce e l'individuazione dei Design Patterns. Inoltre, verranno definiti i packages, le interfacce delle classi e i diagrammi delle classi che riguardano importanti decisioni implementative.

1.1 Trade-off

1.1.1 Comprensibilità vs costi

Una qualità principale che deve avere un sistema software è la comprensibilità del sistema stesso al fine di rendere il codice comprensibile non solo a chi l'ha realizzato ma anche a coloro che sono esterni al progetto o che magari non sono stati coinvolti in una determinata parte del codice. Per rendere il codice quanto più comprensibile verranno utilizzati dei commenti che potranno permettere una maggiore leggibilità al fine di effettuare eventuali e future modifiche o per il mantenimento del sistema. Verranno rispettate inoltre linee guida per l'implementazione del sistema. Daremo quindi maggiore importanza alla comprensibilità anche se ciò comporta un aumento dei costi di sviluppo e di tempo.

1.1.2 Tempo di risposta vs Affidabilità

Il sistema sarà implementato in modo tale da preferire l'affidabilità al tempo di risposta. Per garantire una maggiore robustezza, verrà implementato un controllo più accurato dei dati in input, sia lato utente, utilizzando sia script Javascript, sia tramite controlli lato Servlet, a discapito del tempo di risposta del sistema.

1.1.3 Manutenibilità vs efficienza

In previsione di future manutenzioni evolutive, nello sviluppo del sistema abbiamo preferito avere una maggiore manutenibilità facendo in modo che ogni sottosistema non acceda direttamente al DB, ma che piuttosto l'accesso ai dati venga gestito da un sottosistema intermedio.

1.1 Componenti off-the-shelf

Per il progetto software che si vuole realizzare facciamo uso di componenti ***off-the-shelf,*** che sono componenti software disponibili sul mercato per facilitare la creazione del software.
Il framework che andremo ad utilizzare per il comparto grafico è Bootstrap, che è un framework open source che contiene una raccolta di strumenti liberi per la creazione di siti e applicazioni per il Web. Esso contiene modelli di progettazione basati su HTML e CSS, sia per la tipografia, che per le varie componenti dell'interfaccia, come moduli, bottoni e navigazione, e altri componenti dell'interfaccia, così come alcune estensioni opzionali di JavaScript. Inoltre, verrà utilizzato per gestire le connessioni il connettore JDBC. Risulta senza alcun'ombra di dubbio, più efficiente e veloce utilizzare una componente preesistente e fornita da un ente riconosciuto.
Utilizzeremo inoltre il plugin FullCalendar, riferimento https://fullcalendar.io/, per la visualizzazione e gestione del calendario degli appuntamenti.

1.2 Linee guida per la documentazione dell'interfaccia

Nell'implementazione del sistema, i programmatori dovranno attenersi alle linee guida di seguito definite.

1.3.1 Classi e interfacce Java

Nella scrittura di codice per le classi Java ci si atterrà allo standard Google Java nella sua interezza. Tale standard fornisce delle regole da seguire ad esempio ogni metodo ed ogni file possono non essere preceduti da un commento. Potranno esserci, inoltre, commenti e giustificazioni in merito a particolari decisioni o calcoli. La convenzione utilizzata dai team member per quanto riguarda i nomi delle variabili, è la nota lowerCamelCase, che consiste nello scrivere parole composte o frasi unendo tutte le parole tra loro. Quando si codificano classi e interfacce Java, si dovrebbero rispettare le seguenti regole di formattazione:

1. La parentesi graffa aperta "{"si trova alla fine della stessa linea dell'istruzione di dichiarazione.
2. La parentesi graffa chiusa "}" inizia su una nuova riga vuota allo stesso livello di indentazione del nome della classe o dell'interfaccia.

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

Return new MyClass(){
@Override public void method(){
    if(condition()){
         try{
            something();
         } catch(ProblemException e){
           recover();
         }
      } else if (otherCondition()){
       somethingElse();
    } else {
       lastThing();
  }
 }
};
```

Nel caso di istruzioni semplici, ogni linea deve contenere al massimo una sola istruzione. Mentre nel caso di istruzioni composte vanno rispettate le seguenti regole:

1. Le istruzioni racchiuse all'interno di un blocco (esempio: for), devono essere indentate di un'unità all'interno dell'istruzione composta.
2. La parentesi di apertura del blocco deve trovarsi alla fine della riga dell'istruzione composta.
3. La parentesi di chiusura del blocco deve trovarsi allo stesso livello di indentazione dell'istruzione composta

I nomi di classe devono essere sostantivi, con lettere minuscole e, sia la prima lettera del nome della classe sia la prima lettera di ogni parola interna, deve essere maiuscola. I nomi delle classi dovrebbero essere semplici, descrittivi e che rispettino il dominio applicativo. Non dovrebbero essere usati underscore per legare nomi. I nomi dei metodi iniziano con una lettera minuscola (non

sono consentiti caratteri speciali) e seguono la notazione a cammello. Dovranno essere semplici, descrittivi e che rispettino il dominio applicativo.

Organizzazione dei file

- Ogni file deve essere sviluppato e diviso in base alla categoria di appartenenza, ovvero deve essere correlato ad un'unica funzionalità che persegue. Ad esempio, ogni pagina della Tutoring_Request (compilazione richiesta, visualizzazione stato richiesta, modifica prenotazione, etc.) deve essere implementata in file separati;
- La convenzione per quanto riguarda i nomi dei file, delle operazioni e delle variabili è quella di avere nomi evocativi, ma soprattutto in lingua inglese.
- Organizzare in una cartella i file delle librerie usate e le altre risorse scaricate necessarie per lo sviluppo del progetto.

1.3.2 Pagine lato Server (JSP)

Le pagine JSP devono, quando eseguite, produrre un documento conforme allo standard HTML 5. Il codice Java delle pagine deve aderire alle convenzioni per la codifica in Java, con le seguenti puntualizzazioni:

1. Il tag di apertura (<%) si trova all'inizio di una riga;
2. Il tag di chiusura (%>) si trova all'inizio di una riga;
3. È possibile evitare le due regole precedenti, se il corpo del codice Java consiste in una singola istruzione (<%=%>):

```
<!-Accettabile ->
<% for(String par: paragraphs) {%>
<p class='item'><% out.print(par); %></p>
<%}%>

<!-Non Accettabile->
<p class='item'><% List<String> paragraphs = getParagraphs();
out.print(paragraphs.get(i++));%></p>
```

Per le Servlet è necessario far terminare il nome della classe con il suffisso Servlet.

1.3.3 Pagine HTML

Le pagine HTML, sia in forma statica che dinamica, devono essere conformi allo standard HTML 5. Inoltre, il codice HTML statico deve utilizzare l'indentazione, per facilitare la lettura, secondo le seguenti regole:

1. Un'indentazione consiste in una tabulazione;
2. Ogni tag deve avere un'indentazione maggiore del tag che lo contiene;
3. Ogni tag di chiusura deve avere lo stesso livello di indentazione del corrispondente tag di apertura;
4. I tag di commento devono seguire le stesse regole che si applicano ai tag normali.

Es.

```
<!-Accettabile->
<div>
    <span>
        <ul>
            <li>
                Uno
            </li>
            <li>
                Due
            </li>
        </ul>
    </span>
</div>

<!-Non Accettabile ->
<div><span>
<nl>
<li>Uno</li>
<li>
Due
</li>
</nl></span>
</div>
```

1.3.5 Fogli di stile CSS

I fogli di stile (CSS) devono seguire le seguenti convenzioni:
Ogni regola CSS deve essere formattata come segue:

1. I selettori della regola si trovano a livello 0 di indentazione, uno per riga;
2. L'ultimo selettore della regola è seguito da parentesi graffa aperta ({);
3. Le proprietà che costituiscono la regola sono listate una per riga e sono indentate rispetto ai selettori;
4. La regola è terminata da una parentesi graffa chiusa (}), collocata da sola su una riga;

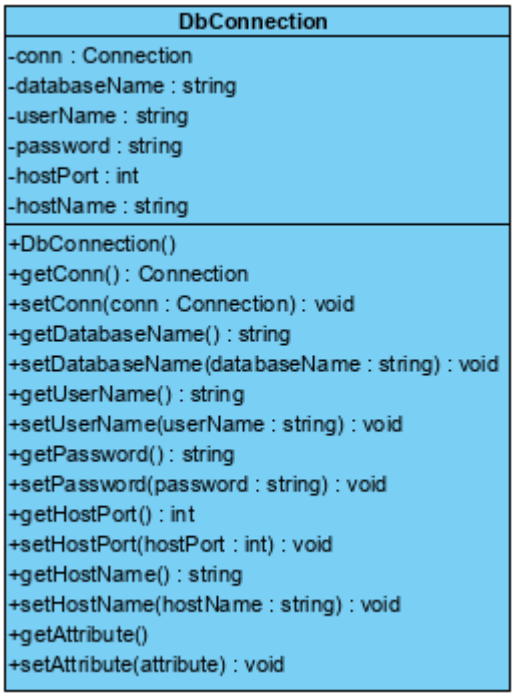
1.3.6 Database SQL

I nomi delle tabelle devono seguire le seguenti regole:
1. Devono essere costituiti da sole lettere maiuscole;
2. Se il nome è costituito da più parole, è previsto l'uso di underscore (_);
3. Il nome deve essere un sostantivo singolare tratto dal dominio del problema ed esplicativo del contenuto.
I nomi dei campi devono seguire le seguenti regole:
1. Devono essere costituiti da sole lettere minuscole e, sia la prima lettera del nome del campo, sia la prima lettera di ogni parola interna, deve essere maiuscola;
2. Il nome deve essere un sostantivo singolare tratto dal dominio del problema ed esplicativo del contenuto.

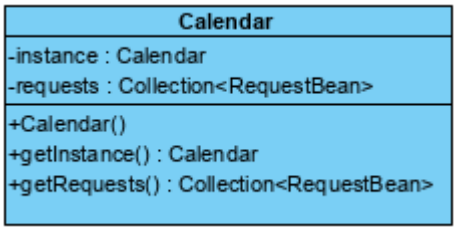
1.3 Design pattern

1.4.1 Singleton

Abbiamo progettato una singola classe (DBConnection) che consentisse di effettuare tutte le operazioni con il database. Per evitare la perdita di efficienza dovuta alla creazione di più istanze di questa classe si è deciso di renderla un Singleton.



Abbiamo progettato una singola classe, Calendar, avente il compito di recuperare le informazioni inerenti alle richieste di appuntamento e permetterne la corretta visualizzazione. Per evitare ridondanze e/o mancata coerenza tra più istanze si è deciso di sviluppare questa classe come Singleton.



1.4 Definizioni, acronimi e abbreviazioni

DBMS: DataBase Management System.

Off-The-Shelf: Servizi esterni di cui viene fatto utilizzo da terzi.
Framework: Software di supporto allo sviluppo web.
HTML: Linguaggio di mark-up per pagine web.
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, di effetti dinamici interattivi tramite funzioni di script invocate da *eventi* innescati a loro volta in vari modi dall'utente sulla pagina web in uso

Bootstrap: Framework che contiene librerie utili per lo sviluppo responsive di pagine web.
Camel Notation: Consiste nello scrivere più parole insieme delimitando la fine e l'inizio di una nuova parola con una lettera maiuscola.
Hard Coding: Codifica fissa.

1.5 Riferimenti

- Ian Sommerville, Software Engineering, Addison Wesely
- TS_SDD_V_0.7.
- http://getbootstrap.com/

2. Packages

2.1 View

Il package View è formato a sua volta da tre packages, Student, Tutor e Commission; inoltre sono presenti le pagine Home JSP, Index JSP, Login JSP, RefuseAccess JSP e Registration JSP.

Il package Student viene implementato con le pagine:

- Calendar JSP: mostra un calendario che consente allo studente di richiedere un appuntamento;
- Request JSP: mostra un'interfaccia grafica che consente allo studente di inserire le informazioni per richiedere un appuntamento;
- RequestInfo JSP: pagina che mostra le informazioni relative ad una richiesta di appuntamento effettuata indicandone lo stato ed i dettagli;
- RequestModify JSP: pagina che consente allo studente di modificare il commento, il giorno e l'orario di un appuntamento richiesto;
- RequestsList JSP: pagina che elenca tutte le richieste di appuntamento effettuate;

Il package Tutor viene implementato con le pagine:

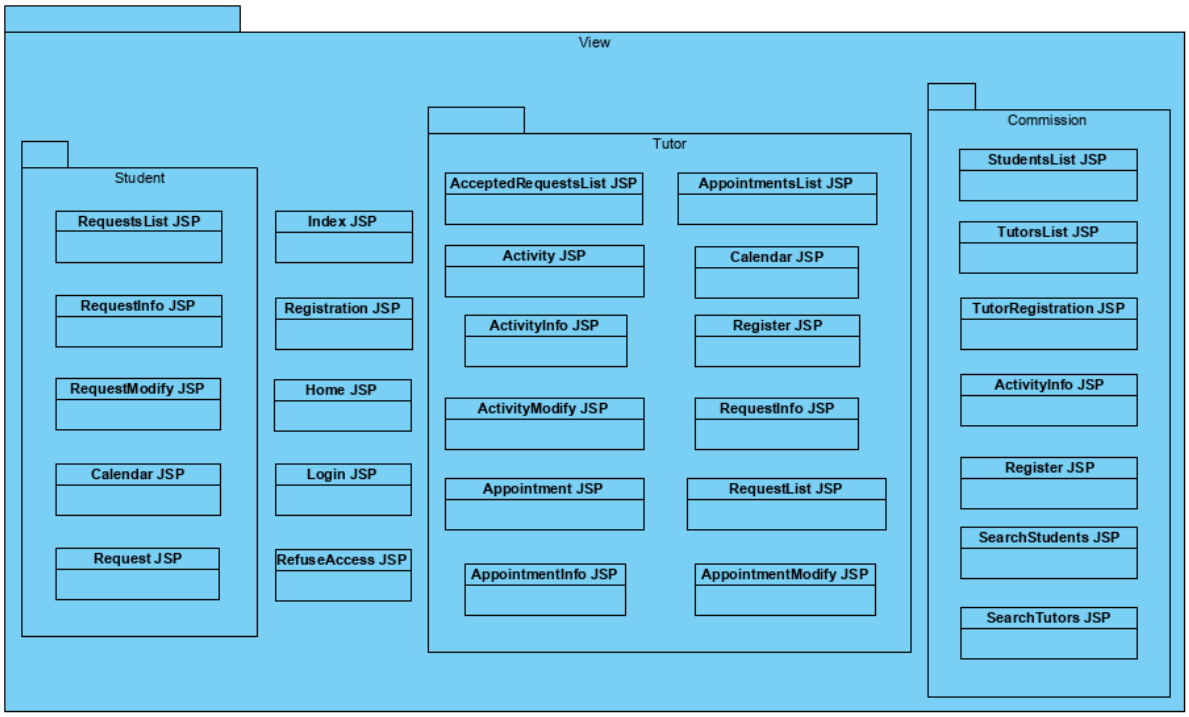
- AcceptedRequestsList JSP: pagina che consente la visualizzazione delle richieste da valutare;
- Activity JSP: pagina che consente la registrazione di un'attività lavorativa svolta;
- ActivityInfo JSP: pagina che mostra le informazioni relative ad un'attività lavorativa registrata;
- ActivityModify JSP: pagina che consente al tutor di modificare i dati di un'attività lavorativa salvata sul registro;
- Appointment JSP: pagina che consente la registrazione di un appuntamento;
- AppointmentInfo JSP: pagina che mostra le informazioni relative ad un appuntamento;
- AppointmentModify JSP: pagina che consente al tutor di modificare i dati di un appuntamento salvato;
- AppointmentsList JSP: pagina che elenca tutti gli appuntamenti confermati dal tutor;
- Calendar JSP: pagina che mostra un calendario per la visualizzazione degli appuntamenti in agenda;
- Register JSP: pagina che mostra le informazioni relative al registro di tutorato;
- RequestInfo JSP: pagina che mostra le informazioni relative ad una richiesta di appuntamento effettuata indicandone lo stato e i dettagli.

- RequestsList JSP: pagina che mostra le richieste di appuntamento consentendo al tutor di valutarle.

Il package Commission viene implementato con le pagine:

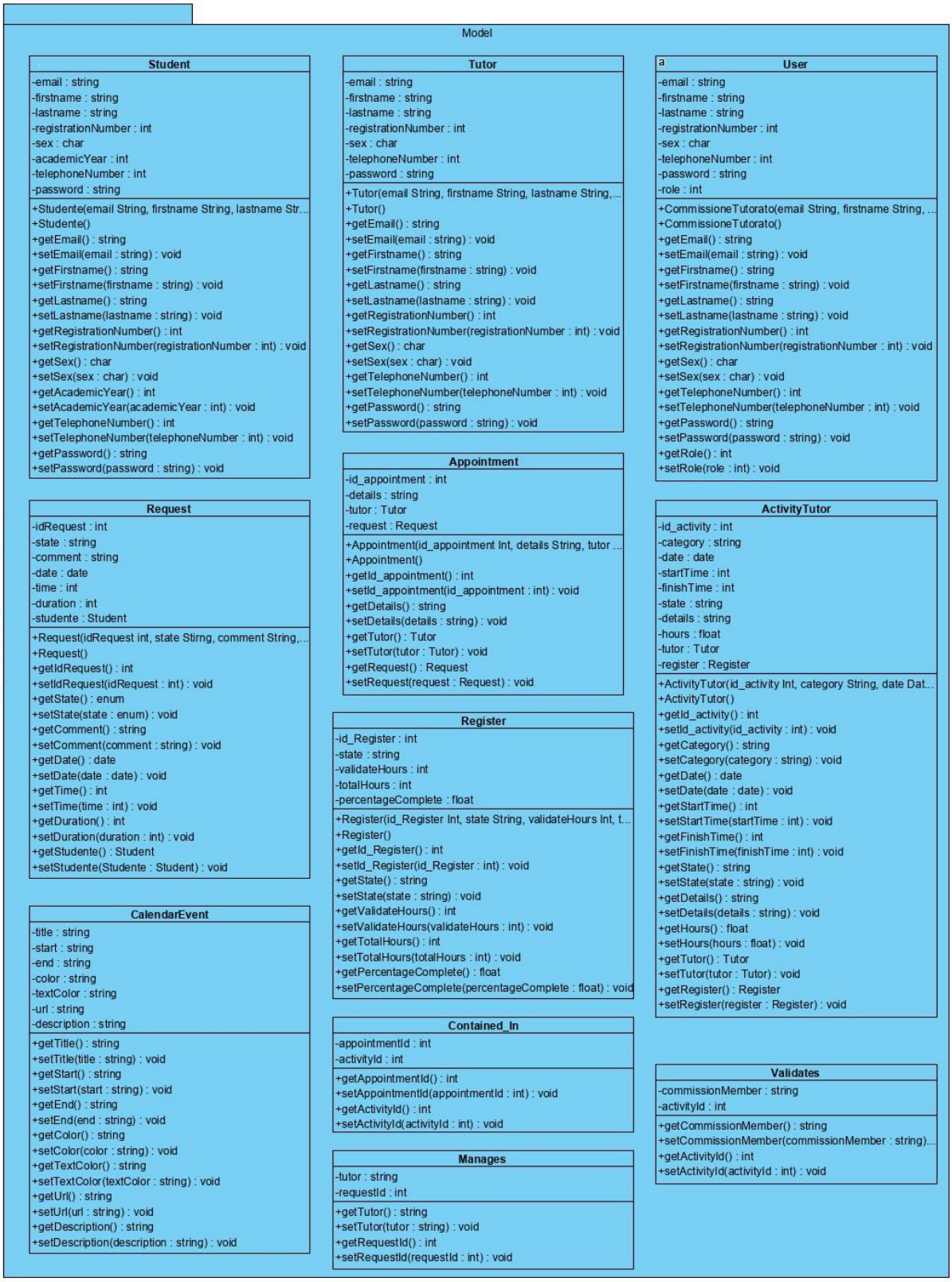
- ActivityInfo JSP: pagina che mostra le informazioni relative ad un'attività di tutorato registrata, consentendone la convalida;
- Register JSP: pagina che mostra le informazioni relative al registro di tutorato di un tutor;
- SearchStudents JSP: pagina che consente la ricerca degli studenti che hanno richiesto un appuntamento in un periodo di tempo;
- SearchTutors JSP: pagina che consente la ricerca dei tutor attivi in un periodo di tempo;
- StudentsList JSP: pagina che elenca tutti gli studenti trovati ed i dettagli degli appuntamenti;
- TutorsList JSP: pagina che elenca tutti i tutor trovati;
- TutorRegistration JSP: pagina che consente la registrazione di un nuovo tutor;

La pagina Login JSP viene implementata per gestire l'autenticazione alla piattaforma, Home JSP è la pagina a cui vengono reindirizzati gli utenti autenticati, mentre Index JSP è la prima pagina che viene mostrata all'utente quando accede alla piattaforma. RefuseAccess JSP è la pagina che viene visualizzata quando si tenta un accesso non consentito ad aree riservate, mentre Registration JSP permette ad un nuovo utente di registrarsi alla piattaforma.



2.2 Model

Il package Model contiene tutte le classi dedite alla gestione dei dati persistenti. Esso si occupa di fare da tramite tra l'applicazione e il database sottostante. Ogni classe contenuta all'interno di questo pacchetto fornisce i metodi per accedere ai dati utili all'applicazione. Le classi contenute all'interno di questo package sono: ActivityTutor, Appointment, CalendarEvent, Contained_In, Manages, Register, Request, Student, Tutor, User, Validates.


Model
Student
-email : string
-firstname : string
-lastname : string
-registrationNumber : int
-sex : char
-academicYear : int
-telephoneNumber : int
-password : string
+Studente(email String, firstname String, lastname Str...
+Studente()
+getEmail() : string
+setEmail(email : string) : void
+getFirstname() : string
+setFirstname(firstname : string) : void
+getLastname() : string
+setLastname(lastname : string) : void
+getRegistrationNumber() : int
+setRegistrationNumber(registrationNumber : int) : void
+getSex() : char
+setSex(sex : char) : void
+getAcademicYear() : int
+setAcademicYear(academicYear : int) : void
+getTelephoneNumber() : int
+setTelephoneNumber(telephoneNumber : int) : void
+getPassword() : string
+setPassword(password : string) : void
Tutor
-email : string
-firstname : string
-lastname : string
-registrationNumber : int
-sex : char
-telephoneNumber : int
-password : string
+Tutor(email String, firstname String, lastname String,...
+Tutor()
+getEmail() : string
+setEmail(email : string) : void
+getFirstname() : string
+setFirstname(firstname : string) : void
+getLastname() : string
+setLastname(lastname : string) : void
+getRegistrationNumber() : int
+setRegistrationNumber(registrationNumber : int) : void
+getSex() : char
+setSex(sex : char) : void
+getTelephoneNumber() : int
+setTelephoneNumber(telephoneNumber : int) : void
+getPassword() : string
+setPassword(password : string) : void
a
User
-email : string
-firstname : string
-lastname : string
-registrationNumber : int
-sex : char
-telephoneNumber : int
-password : string
-role : int
+CommissioneTutorato(email String, firstname String, ...
+CommissioneTutorato()
+getEmail() : string
+setEmail(email : string) : void
+getFirstname() : string
+setFirstname(firstname : string) : void
+getLastname() : string
+setLastname(lastname : string) : void
+getRegistrationNumber() : int
+setRegistrationNumber(registrationNumber : int) : void
+getSex() : char
+setSex(sex : char) : void
+getTelephoneNumber() : int
+setTelephoneNumber(telephoneNumber : int) : void
+getPassword() : string
+setPassword(password : string) : void
+getRole() : int
+setRole(role : int) : void
Request
-idRequest : int
-state : string
-comment : string
-date : date
-time : int
-duration : int
-studente : Student
+Request(idRequest int, state Stirng, comment String,...
+Request()
+getIdRequest() : int
+setIdRequest(idRequest : int) : void
+getState() : enum
+setState(state : enum) : void
+getComment() : string
+setComment(comment : string) : void
+getDate() : date
+setDate(date : date) : void
+getTime() : int
+setTime(time : int) : void
+getDuration() : int
+setDuration(duration : int) : void
+getStudente() : Student
+setStudente(Studente : Student) : void
Appointment
-id_appointment : int
-details : string
-tutor : Tutor
-request : Request
+Appointment(id_appointment Int, details String, tutor ...
+Appointment()
+getId_appointment() : int
+setId_appointment(id_appointment : int) : void
+getDetails() : string
+setDetails(details : string) : void
+getTutor() : Tutor
+setTutor(tutor : Tutor) : void
+getRequest() : Request
+setRequest(request : Request) : void
ActivityTutor
-id_activity : int
-category : string
-date : date
-startTime : int
-finishTime : int
-state : string
-details : string
-hours : float
-tutor : Tutor
-register : Register
+ActivityTutor(id_activity Int, category String, date Dat...
+ActivityTutor()
+getId_activity() : int
+setId_activity(id_activity : int) : void
+getCategory() : string
+setCategory(category : string) : void
+getDate() : date
+setDate(date : date) : void
+getStartTime() : int
+setStartTime(startTime : int) : void
+getFinishTime() : int
+setFinishTime(finishTime : int) : void
+getState() : string
+setState(state : string) : void
+getDetails() : string
+setDetails(details : string) : void
+getHours() : float
+setHours(hours : float) : void
+getTutor() : Tutor
+setTutor(tutor : Tutor) : void
+getRegister() : Register
+setRegister(register : Register) : void
Register
-id_Register : int
-state : string
-validateHours : int
-totalHours : int
-percentageComplete : float
+Register(id_Register Int, state String, validateHours Int, t...
+Register()
+getId_Register() : int
+setId_Register(id_Register : int) : void
+getState() : string
+setState(state : string) : void
+getValidateHours() : int
+setValidateHours(validateHours : int) : void
+getTotalHours() : int
+setTotalHours(totalHours : int) : void
+getPercentageComplete() : float
+setPercentageComplete(percentageComplete : float) : void
CalendarEvent
-title : string
-start : string
-end : string
-color : string
-textColor : string
-url : string
-description : string
+getTitle() : string
+setTitle(title : string) : void
+getStart() : string
+setStart(start : string) : void
+getEnd() : string
+setEnd(end : string) : void
+getColor() : string
+setColor(color : string) : void
+getTextColor() : string
+setTextColor(textColor : string) : void
+getUrl() : string
+setUrl(url : string) : void
+getDescription() : string
+setDescription(description : string) : void
Contained_In
-appointmentId : int
-activityId : int
+getAppointmentId() : int
+setAppointmentId(appointmentId : int) : void
+getActivityId() : int
+setActivityId(activityId : int) : void
Validates
-commissionMember : string
-activityId : int
+getCommissionMember() : string
+setCommissionMember(commissionMember : string)...
+getActivityId() : int
+setActivityId(activityId : int) : void
Manages
-tutor : string
-requestId : int
+getTutor() : string
+setTutor(tutor : string) : void
+getRequestId() : int
+setRequestId(requestId : int) : void

2.3 Control

Il package Control riceve, tramite il pacchetto View, i comandi dell'utente.
Esso è composta da 6 packages, Request_Management, Security, Tutoring_Activity_Management, Tutoring_Request, Tutoring_Supervision, User.

Il package Request_Management viene implementato con le servlet:

- AppointmentServlet: si occupa di gestire la registrazione, modifica e cancellazione di un appuntamento;
- CalendarServlet: permette al sistema di ricercare le richieste di appuntamento e visualizzarle in un calendario con annesso collegamento ai dettagli ed inoltre indica gli orari di attività dello sportello;
- RequestServlet: gestisce l'accettazione delle richieste di appuntamento, e l'aggiornamento dello stato qualora lo studente risulti assente;
- ShowAppointmentServlet: gestisce la visualizzazione dei dettagli di un appuntamento;
- ShowRequestServlet: si occupa di gestire la visualizzazione dei dettagli di una richiesta di appuntamento.

Il package Security viene implementato con le classi:

- AdminFilter: verifica che l'utente che sta provando ad accedere alle pagine dell'area Commissione sia effettivamente un membro della Commissione di Tutorato autenticato;
- AuthFilter: controlla che l'utente abbia effettuato l'accesso prima di accedere alle pagine private;
- StudentFilter: verifica che l'utente che sta provando ad accedere alle pagine dell'area studente sia uno studente autenticato;
- TutorFilter: controlla che l'utente che sta provando ad accedere alle pagine dell'area tutor sia un tutor autenticato.

Il package Tutoring_Activity_Management viene implementato con le servlet:

- ActivityServlet: si occupa di gestire la registrazione, modifica e cancellazione di un'attività di tutorato;
- ShowActivityServlet: gestisce la visualizzazione dei dettagli di un'attività di tutorato;
- ShowRegisterServlet: gestisce la visualizzazione dei dettagli del registro del tutor;

Il package Tutoring_Request viene implementato con le servlet:

- CalendarServlet: permette al sistema di ricercare le richieste di appuntamento e visualizzarle in un calendario, che indica inoltre gli orari di attività dello sportello;
- RequestServlet: gestisce la registrazione, modifica e cancellazione delle richieste di appuntamento;
- ShowRequestServlet: si occupa di gestire la visualizzazione dei dettagli di una richiesta di appuntamento.

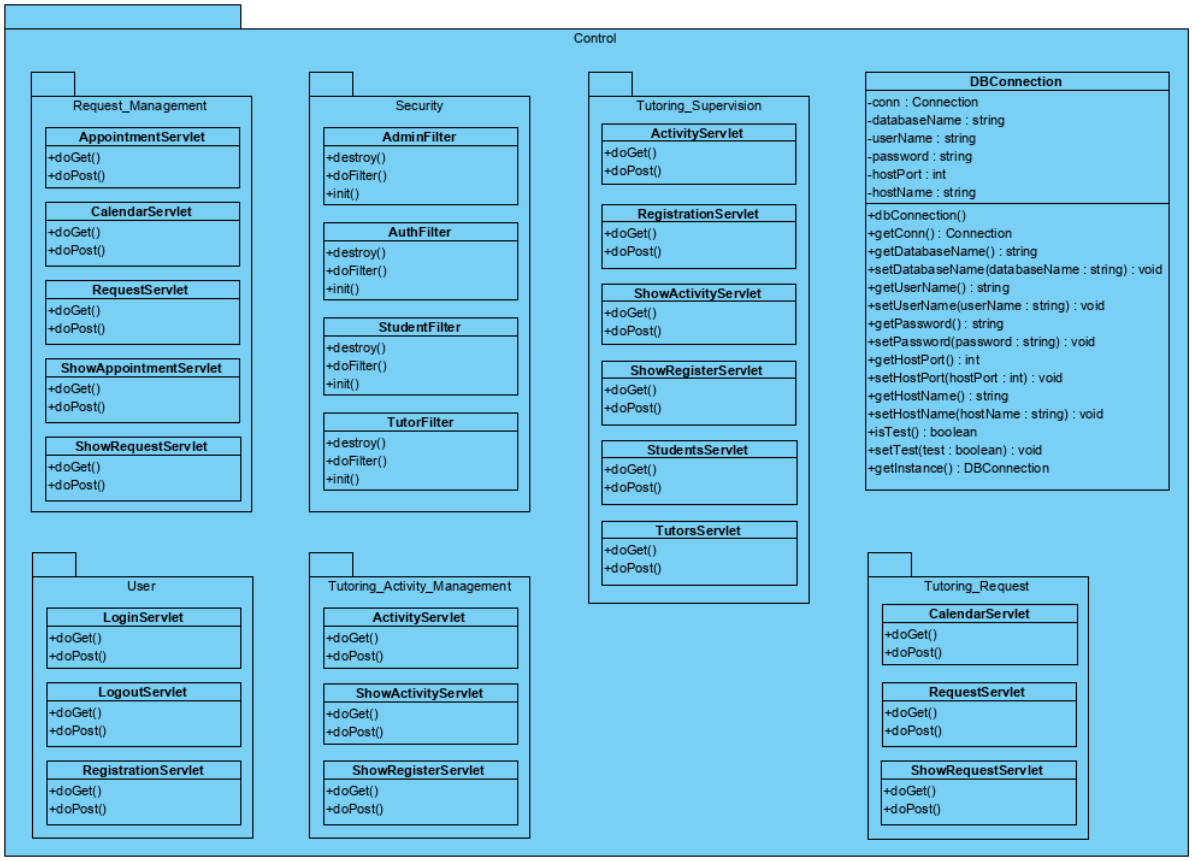
Il package Tutoring_Supervision viene implementato con le servlet:

- ActivityServlet: si occupa di gestire la convalida e rimozione delle attività di tutorato registrate dai tutor;
- RegistrationServlet: si occupa di gestire la registrazione di un nuovo tutor;
- ShowActivityServlet: gestisce la visualizzazione dei dettagli di un'attività di tutorato;
- ShowRegisterServlet: gestisce la visualizzazione dei dettagli del registro di un tutor;
- StudentsServlet: si occupa di gestire la ricerca degli studenti e di visualizzarli in un elenco;
- TutorsServlet: si occupa di gestire la ricerca dei tutor e di visualizzarli in un elenco;

Il package User viene implementato con le servlet:

- LoginServlet: gestisce l'accesso degli utenti alla piattaforma;
- LogoutServlet: gestisce il logout degli utenti dalla piattaforma;
- RegistrationServlet: si occupa di gestire la registrazione di un nuovo utente;

Infine all'interno del pacchetto è presente la classe DBConnection che si occupa di gestire l'intero Database, e comunica con tutte le Servlet presenti in questo pacchetto.



3. Interfacce delle classi

Nome classe	Tutoring_Request
Descrizione	Rappresenta il gestore delle funzionalità relate alle richieste di tutorato.
Pre-condizione	**context** Tutoring_Request:: addRequest(Data); **pre:** Data!=null && isValid(Data) **context** Tutoring_Request:: modifyRequest(Request, Data); **pre:** Request!=null && exists(Request) && Data!=null && isValid(Data) **context** Tutoring_Request:: showRequest(Request); **pre:** Request!=null && exists(Request)
Post-condizione	**context** Tutoring_Request:: addRequest(Data); **post:** exists(Request)
Invarianti	

Nome classe	Request_Management
Descrizione	Rappresenta il gestore delle funzionalità relate alla gestione delle richieste.
Pre-condizione	**context** Request_Management:: addAppointment(Data); **pre:** Data!=null && isValid(Data) **context** Request_Management:: getStudentData(Student); **pre:** Student!=null && exists(Student) **context** Request_Management:: modifyRequest(Request); **pre:** Request!=null && exists(Request) **context** Request_Management:: showRequest(Request); **pre:** Request!=null && exists(Request)
Post-condizione	**context** Request_Management:: addAppointment(Data); **post:** exists(Appointment)
Invarianti	

Nome classe	Tutoring_Activity_Management
Descrizione	Rappresenta il gestore delle funzionalità relate alla gestione dell'attività di tutorato.
Pre-condizione	**context** Tutoring_Activity_Management:: addActivity(Data); **pre:** Data!=null && isValid(Data) **context** Tutoring_Activity_Management:: generateRegister(Register); **pre:** Register!=null && exists(Register) **context** Tutoring_Activity_Management:: getActivityData(Activity); **pre:** Activity!=null && exists(Activity)

	context Tutoring_Activity_Management:: getAppointmentData(Appointment); **pre:** Appointment!=null && exists(Appointment) **context** Tutoring_Activity_Management:: modifyActivity(Activity, Data); **pre:** Activity!=null && exists(Activity) && Data!=null && isValid(Data) **context** Tutoring_Activity_Management:: modifyAppointment(Appointment, Data); **pre:** Appointment!=null && exists(Appointment) && Data!=null && isValid(Data) **context** Tutoring_Activity_Management:: showActivity(Activity); **pre:** Activity!=null && exists(Activity) **context** Tutoring_Activity_Management:: showAppointment(Appointment); **pre:** Appointment!=null && exists(Appointment) **context** Tutoring_Activity_Management:: showCalendar();
Post-condizione	
Invarianti	

Nome classe	Tutoring_Supervision
Descrizione	Rappresenta il gestore delle funzionalità relate alla supervisione dell'attività di tutorato.
Pre-condizione	**context** Tutoring_Supervision:: addTutor(Data); **pre:** Data!=null && isValid(Data) **context** Tutoring_Supervision :: modifyWorkDay(WorkDay, Data); **pre:** WorkDay!=null && exists(WorkDay) && Data!=null && isValid(Data) **context** Tutoring_Supervision :: modifyActivity(Activity, Data); **pre:** Activity!=null && exists(Activity) && Data!=null && isValid(Data) **context** Tutoring_Supervision :: searchTutors(Data); **pre:** Data!=null && isValid(Data) **context** Tutoring_Supervision :: showRegister(Register); **pre:** Register!=null && exists(Register) **context** Tutoring_Supervision :: validateActivity(Activity); **pre:** Activity!=null && exists(Activity)
Post-condizione	**context** Tutoring_Supervision :: addTutor(Data); **post:** exists(Tutor)
Invarianti	

4. 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.
FullCalendar: Plugin che consente la visualizzazione e gestione di un calendario.
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.
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.