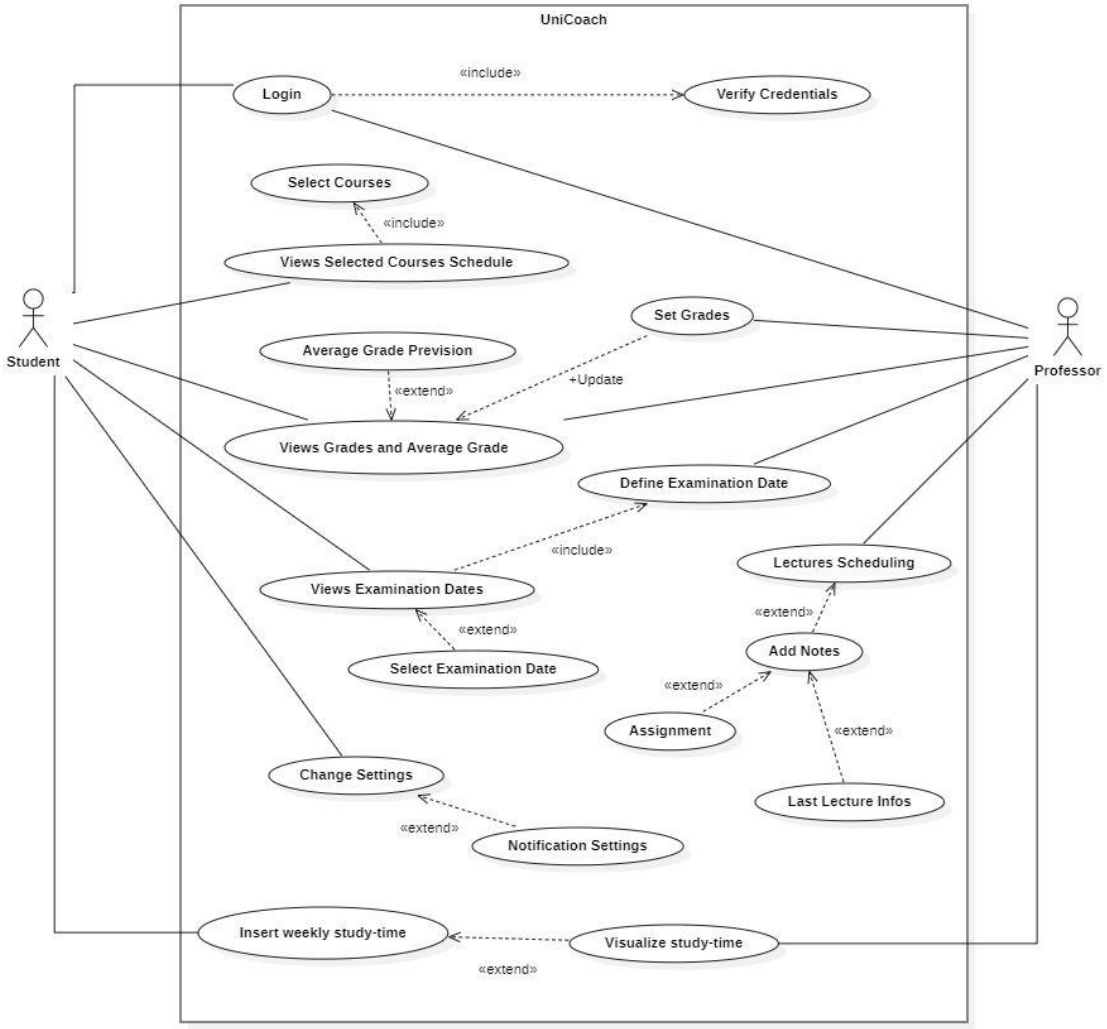
Di seguito illustriamo lo Use Case Diagram realizzato per la nostra applicazione UniCoach.



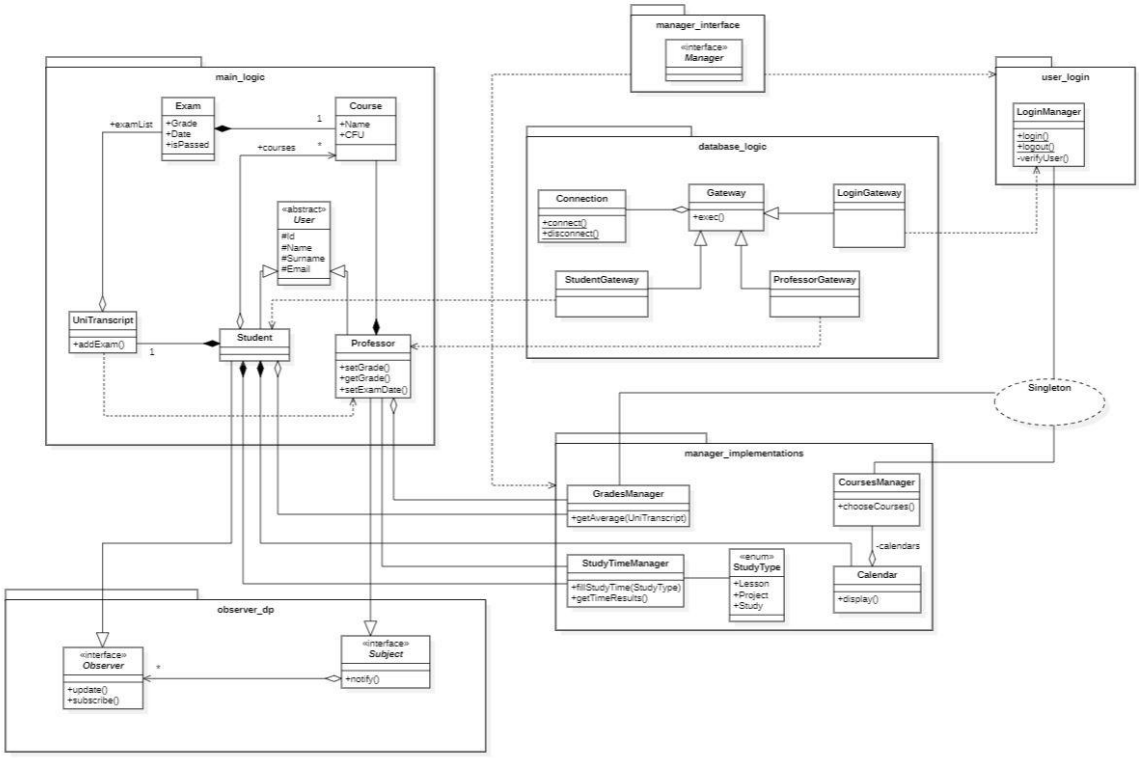
Si individuano due tipologie di utenti principali, Studente e Professore, i quali avranno possibilità di compiere azioni diverse.

Lo studente, una volta effettuato il login con le corrette credenziali, sceglie i corsi che intende seguire durante l'anno e visualizza quindi il calendario settimanale delle lezioni selezionate. Successivamente potrà monitorare la sua situazione attuale, visualizzando media, voti ed eventuali esami da svolgere. Infine abbiamo voluto aggiungere una feature che permetta allo studente di tenere traccia del proprio tempo speso nello studio, potendo quindi classificarlo in ore dedicate alle lezioni, allo svolgimento di progetti, oppure allo studio per l'esame.

Il professore esegue anch'esso l'accesso all'applicazione e successivamente potrà andare a gestire il proprio corso di insegnamento inserendo le date dell'esame, aggiornando i voti degli studenti, impostando gli orari per le lezioni ed eventualmente segnalando ai propri studenti aggiornamenti ed homework specifici. Infine il docente potrà monitorare lo studio dei propri studenti andando a vedere il report che gli stessi hanno inserito sulla piattaforma

indicando come sono state spese le ore di studio della materia, potendo così evidenziare eventuali aggiustamenti e migliorie per il proprio corso.

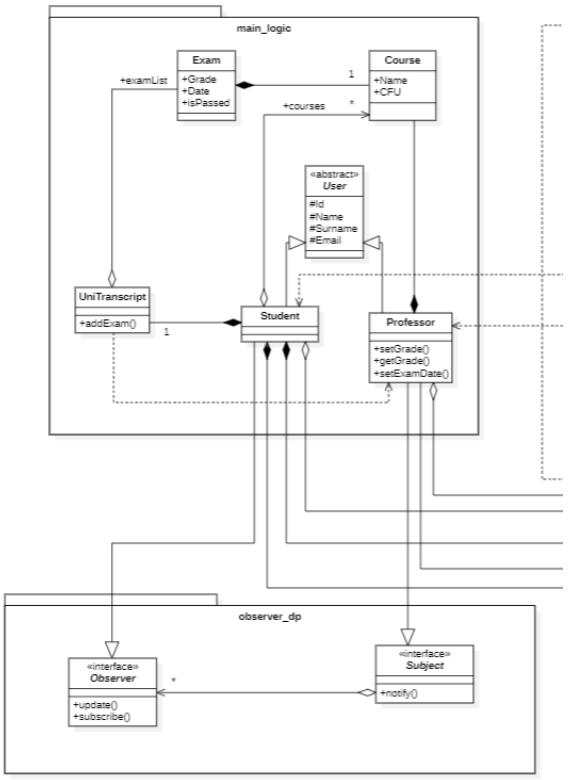
Passando al class diagram:



Andiamo ad analizzare le parti principali:

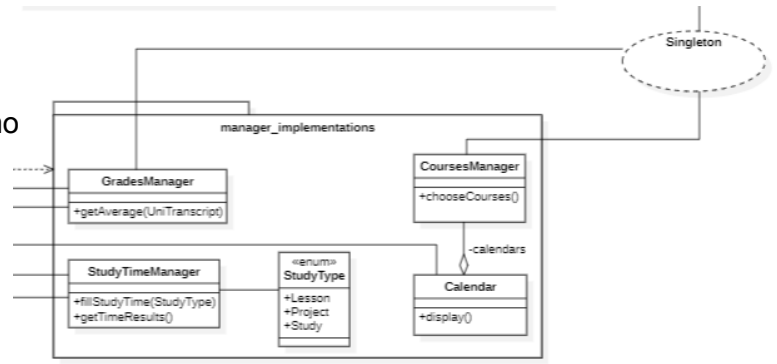
- **Main Logic**

La logica principale del programma si sviluppa partendo dalle classi Student e Professor che svolgono il ruolo di utenti. Abbiamo deciso di implementare il design pattern Observer tra queste due classi in modo da poter notificare lo studente ogni qual volta che il docente inserisce delle note nella lezione appena svolta. Lo studente avrà quindi una reference ad una classe UniTranscript che modella il libretto universitario, dove vengono salvati i risultati degli esami conseguiti (gestiti tramite examList) ed aggiornati direttamente dal professore. Quest'ultimo presenta un collegamento con la classe Course che ci permette di modellare il corso fornito dal docente.



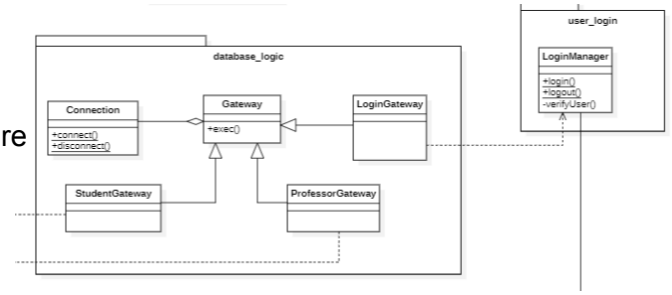
- **Managers**

Abbiamo deciso di implementare delle classi managers che ci permettono di gestire le varie feature dell'applicazione. In particolare abbiamo un manager per ogni task, quali gestione della media, scelta dei corsi da seguire ed infine StudyTimeManager che viene utilizzato per definire come sono state spese le varie ore di studio da parte dello studente, ed è quindi visualizzabile anche da parte del docente.



- **Database Logic**

Abbiamo deciso di utilizzare il design pattern Gateway che ci permette di interfacciare l'applicazione con la parte di database tramite delle classi apposite che sono dedicate ad eseguire le varie query. Infine la classe LoginManager è dedicata alla verifica dell'utente.



Infine abbiamo realizzato una bozza per lo schema ER che ci permette poi di realizzare il database: