

**Università degli Studi di Napoli Federico II**

**Corso di Laurea in Ingegneria Informatica**

**Corso di Ingegneria del Software**

**Prof. A.R. Fasolino - A.A. 2024 - 25**

****

**“TASK4CLASS”**

**STUDENTI:**

COZZOLINO DARIA N46007242

DI SOMMA DOMENICO N46007119

CAUSA GIOVANNI N46005910

**INDICE**

[**1. Specifiche informali**](https://docs.google.com/document/d/10P17CnIIevG6XvvbWla-IMPJ45-6QxNMbkSH4n9AOgw/edit#heading=h.koitjge9pafb) 3

[**2. Analisi e specifica dei requisiti**](https://docs.google.com/document/d/10P17CnIIevG6XvvbWla-IMPJ45-6QxNMbkSH4n9AOgw/edit#heading=h.2lyua1l0g049) 4

[2.1 Analisi nomi-verbi](https://docs.google.com/document/d/10P17CnIIevG6XvvbWla-IMPJ45-6QxNMbkSH4n9AOgw/edit#heading=h.4pegk0nippxy) 4

[2.2 Revisione dei requisiti](https://docs.google.com/document/d/10P17CnIIevG6XvvbWla-IMPJ45-6QxNMbkSH4n9AOgw/edit#heading=h.50hjwgwjm6g9) 5

[2.3 Glossario dei termini](https://docs.google.com/document/d/10P17CnIIevG6XvvbWla-IMPJ45-6QxNMbkSH4n9AOgw/edit#heading=h.cv5rwvws6qvi) 7

[2.4 Classificazione dei requisiti](https://docs.google.com/document/d/10P17CnIIevG6XvvbWla-IMPJ45-6QxNMbkSH4n9AOgw/edit#heading=h.6ib3ku2jmuc) 8

[*2.4.1 Requisiti funzionali*](https://docs.google.com/document/d/10P17CnIIevG6XvvbWla-IMPJ45-6QxNMbkSH4n9AOgw/edit#heading=h.x4ejbqkaavfb) 8

[*2.4.2 Requisiti sui dati*](https://docs.google.com/document/d/10P17CnIIevG6XvvbWla-IMPJ45-6QxNMbkSH4n9AOgw/edit#heading=h.d0gaylx9x8ry) 10

[*2.4.3 Vincoli / Altri requisiti*](https://docs.google.com/document/d/10P17CnIIevG6XvvbWla-IMPJ45-6QxNMbkSH4n9AOgw/edit#heading=h.qlu2kjxe5y2n) 11

[2.5 Modellazione dei casi d’uso](https://docs.google.com/document/d/10P17CnIIevG6XvvbWla-IMPJ45-6QxNMbkSH4n9AOgw/edit#heading=h.jf21qn7vpeg1) 13

[*2.5.1 Attori e casi d’uso*](https://docs.google.com/document/d/10P17CnIIevG6XvvbWla-IMPJ45-6QxNMbkSH4n9AOgw/edit#heading=h.tfu6hxxcd6c1) 13

[*2.5.2 Diagramma dei casi d’uso*](https://docs.google.com/document/d/10P17CnIIevG6XvvbWla-IMPJ45-6QxNMbkSH4n9AOgw/edit#heading=h.si4oar3whwuv) 15

[*2.5.3 Scenari*](https://docs.google.com/document/d/10P17CnIIevG6XvvbWla-IMPJ45-6QxNMbkSH4n9AOgw/edit#heading=h.4m1eajhta0zl) 16

[2.6 Diagramma delle classi](https://docs.google.com/document/d/10P17CnIIevG6XvvbWla-IMPJ45-6QxNMbkSH4n9AOgw/edit#heading=h.3x5r6v4cwdwg) 25

[*2.6.1 Responsabilità*](https://docs.google.com/document/d/10P17CnIIevG6XvvbWla-IMPJ45-6QxNMbkSH4n9AOgw/edit#heading=h.ulloyxkmrh0a) 26

[2.7 Diagrammi di sequenza](https://docs.google.com/document/d/10P17CnIIevG6XvvbWla-IMPJ45-6QxNMbkSH4n9AOgw/edit#heading=h.10anrti7s8db) 28

[*2.7.1 Iscrizione Classe Studente*](https://docs.google.com/document/d/10P17CnIIevG6XvvbWla-IMPJ45-6QxNMbkSH4n9AOgw/edit#heading=h.1mnd7p2p9kl4) 28

[*2.7.3 Creazione Task*](https://docs.google.com/document/d/10P17CnIIevG6XvvbWla-IMPJ45-6QxNMbkSH4n9AOgw/edit#heading=h.1mnd7p2p9kl4) 29

[*2.7.4 Carica Soluzione*](https://docs.google.com/document/d/10P17CnIIevG6XvvbWla-IMPJ45-6QxNMbkSH4n9AOgw/edit#heading=h.1mnd7p2p9kl4) 30

[**3. Piano di test funzionale**](https://docs.google.com/document/d/10P17CnIIevG6XvvbWla-IMPJ45-6QxNMbkSH4n9AOgw/edit#heading=h.9u31j8f55fpi) 32

*3.1 Creazione Task*32

*3.2 Caria Soluzione*34

*3.3* [*Iscrizione Classe Studente*](https://docs.google.com/document/d/10P17CnIIevG6XvvbWla-IMPJ45-6QxNMbkSH4n9AOgw/edit#heading=h.5mj633elvoht) 35

[**4. Progettazione**](https://docs.google.com/document/d/10P17CnIIevG6XvvbWla-IMPJ45-6QxNMbkSH4n9AOgw/edit#heading=h.46fv4nf3rw5v) 36

[4.1 Diagramma delle classi](https://docs.google.com/document/d/10P17CnIIevG6XvvbWla-IMPJ45-6QxNMbkSH4n9AOgw/edit#heading=h.411jurl6zq4e) 36

[*4.1.1 Traduzione classi ed associazioni*](https://docs.google.com/document/d/10P17CnIIevG6XvvbWla-IMPJ45-6QxNMbkSH4n9AOgw/edit#heading=h.og180cpah5n7) 37

[*4.1.2 Pattern BCED*](https://docs.google.com/document/d/10P17CnIIevG6XvvbWla-IMPJ45-6QxNMbkSH4n9AOgw/edit#heading=h.2hdgvfuw7zj6) 38

[4.1.2.1 Package Boundary](https://docs.google.com/document/d/10P17CnIIevG6XvvbWla-IMPJ45-6QxNMbkSH4n9AOgw/edit#heading=h.5rs1fdibna2m) 38

[4.1.2.2 Package Controller](https://docs.google.com/document/d/10P17CnIIevG6XvvbWla-IMPJ45-6QxNMbkSH4n9AOgw/edit#heading=h.3uqw7bc7drjm) 39

[4.1.2.3 Package Entity](https://docs.google.com/document/d/10P17CnIIevG6XvvbWla-IMPJ45-6QxNMbkSH4n9AOgw/edit#heading=h.657ao3l1x50m) 39

[4.1.2.4 Package Database](https://docs.google.com/document/d/10P17CnIIevG6XvvbWla-IMPJ45-6QxNMbkSH4n9AOgw/edit#heading=h.5ck9b8aa490z) 40

[4.2 Diagrammi di sequenza](https://docs.google.com/document/d/10P17CnIIevG6XvvbWla-IMPJ45-6QxNMbkSH4n9AOgw/edit#heading=h.onnoz9y59sqb) 42

[*4.2.1 Creazione Task*](https://docs.google.com/document/d/10P17CnIIevG6XvvbWla-IMPJ45-6QxNMbkSH4n9AOgw/edit#heading=h.rrwysdsf4dks) 42

[*4.2.2 Iscrizione Classe Studente*](https://docs.google.com/document/d/10P17CnIIevG6XvvbWla-IMPJ45-6QxNMbkSH4n9AOgw/edit#heading=h.3ub8imnoiv2a) 43

[*4.2.3 Carica Soluzione*](https://docs.google.com/document/d/10P17CnIIevG6XvvbWla-IMPJ45-6QxNMbkSH4n9AOgw/edit#heading=h.3ub8imnoiv2a) 44

[4.2 Diagrammi delle Classi Di Progetto](https://docs.google.com/document/d/10P17CnIIevG6XvvbWla-IMPJ45-6QxNMbkSH4n9AOgw/edit#heading=h.onnoz9y59sqb) 45

[**5. Implementazione**](https://docs.google.com/document/d/10P17CnIIevG6XvvbWla-IMPJ45-6QxNMbkSH4n9AOgw/edit#heading=h.m5tf0p9mnp9b) 46

[5.1 Package Database](https://docs.google.com/document/d/10P17CnIIevG6XvvbWla-IMPJ45-6QxNMbkSH4n9AOgw/edit#heading=h.ijmaxx3syv88) 47

[5.2 Package Entity](https://docs.google.com/document/d/10P17CnIIevG6XvvbWla-IMPJ45-6QxNMbkSH4n9AOgw/edit#heading=h.u0abrr9s4alc) 48

[5.3 Package Controller](https://docs.google.com/document/d/10P17CnIIevG6XvvbWla-IMPJ45-6QxNMbkSH4n9AOgw/edit#heading=h.d5maylr26e8v) 49

[5.4 Package Boundary](https://docs.google.com/document/d/10P17CnIIevG6XvvbWla-IMPJ45-6QxNMbkSH4n9AOgw/edit#heading=h.r4shdrv4dtpa) 50

[5.5 Diagramma di Deployment](https://docs.google.com/document/d/10P17CnIIevG6XvvbWla-IMPJ45-6QxNMbkSH4n9AOgw/edit#heading=h.71h6gr9dbfb9) 51

[**6. Testing**](https://docs.google.com/document/d/10P17CnIIevG6XvvbWla-IMPJ45-6QxNMbkSH4n9AOgw/edit#heading=h.w2tluku5taj5) 52

[6.1 Test strutturale](https://docs.google.com/document/d/10P17CnIIevG6XvvbWla-IMPJ45-6QxNMbkSH4n9AOgw/edit#heading=h.w2tluku5taj5) 52

[*6..2 Test di Unità*](https://docs.google.com/document/d/10P17CnIIevG6XvvbWla-IMPJ45-6QxNMbkSH4n9AOgw/edit#heading=h.6tnclwqj25h) 56

[6.3 Test funzionale](https://docs.google.com/document/d/10P17CnIIevG6XvvbWla-IMPJ45-6QxNMbkSH4n9AOgw/edit#heading=h.ph0yupojn6ns) 57

1. **SPECIFICHE INFORMALI**

**Scenario:**

Si desidera sviluppare un sistema software per la gestione di task didattici assegnati da docenti a studenti, con un meccanismo di punteggio che consente di generare classifiche dinamiche basate sui risultati ottenuti dagli studenti.

**Descrizione del Sistema:**

Il sistema consente l’accesso di docenti e studenti a una piattaforma digitale mediante autenticazione tramite credenziali personali. Al momento della registrazione, ciascun utente deve specificare il proprio ruolo (studente o docente), oltre a fornire nome, cognome e indirizzo e-mail istituzionale. Ogni docente ha la possibilità di creare una o più classi virtuali, ciascuna identificata da un nome e da un codice univoco. L’iscrizione di uno studente a una classe può avvenire in due modalità: il docente può iscrivere direttamente lo studente alla propria classe, selezionandolo da un elenco di studenti iscritti alla piattaforma, oppure lo studente stesso può iscriversi autonomamente, inserendo il codice univoco della classe fornito dal docente. Ogni studente può appartenere ad una sola classe (per semplicità).

All’interno di ogni classe, il docente può creare e assegnare task didattici agli studenti. Ogni task è definito da un titolo, una descrizione, una data di scadenza, un numero massimo di punti assegnabili. Gli studenti possono visualizzare l’elenco dei task assegnati direttamente all’interno del proprio profilo personale. Una volta selezionato un task, possono procedere con la consegna caricando la propria soluzione, che dovrà essere fornita in formato testuale attraverso un’apposita interfaccia. Il docente ha la facoltà di valutare ogni consegna assegnando un punteggio compreso tra zero e il massimo indicato nel task.

Il sistema mantiene lo storico di tutti i task svolti da ciascuno studente e i relativi punteggi ottenuti.

Ogni studente può visualizzare la classifica relativa esclusivamente alla propria classe, attraverso un’apposita sezione del profilo. Le classifiche disponibili sono due: una basata sul punteggio totale accumulato e una basata sul numero di task completati. Queste classifiche consentono agli studenti di confrontare le proprie performance con quelle degli altri membri della stessa classe, incentivando la partecipazione attiva alle attività didattiche. Per semplicità si può supporre che la classifica mostri solo un numero limitato di studente (ex. i primi tre, o i primi cinque)

Ogni studente ha accesso a un profilo personale dove può consultare il totale dei task assegnati, il punteggio totale ottenuto, e la media dei voti ottenuti. I docenti, attraverso un’interfaccia dedicata, possono visualizzare l’insieme delle classi create e, per ciascuna classe, visualizzare l’elenco degli studenti iscritti e l’elenco dei task assegnati.

Il sistema dovrà essere accessibile via web sia da desktop che da dispositivi mobili, con un’interfaccia grafica intuitiva e responsiva. È previsto un sistema di notifiche che avvisi gli studenti della pubblicazione di nuovi task e dell’avvenuta correzione delle proprie consegne.

**2. ANALISI E SPECIFICA DEI REQUISITI**

**2.1 Analisi nomi verbi**

**Descrizione del Sistema:**

Il sistema consente l’accesso di docenti e studenti a una piattaforma digitale mediante autenticazione tramite credenziali personali. Al momento della registrazione, ciascun utente deve specificare il proprio ruolo (studente o docente), oltre a fornire nome, cognome e indirizzo e-mail istituzionale. Ogni docente ha la possibilità di creare una o più classi virtuali, ciascuna identificata da un nome e da un codice univoco. L’iscrizione di uno studente a una classe può avvenire in due modalità: il docente può iscrivere direttamente lo studente alla propria classe, selezionandolo da un elenco di studenti iscritti alla piattaforma, oppure lo studente stesso può iscriversi autonomamente, inserendo il codice univoco della classe fornito dal docente. Ogni studente può appartenere ad una sola classe (per semplicità).

All’interno di ogni classe, il docente può creare e assegnare task didattici agli studenti. Ogni task è definito da un titolo, una descrizione, una data di scadenza, un numero massimo di punti assegnabili. Gli studenti possono visualizzare l’elenco dei task assegnati direttamente all’interno del proprio profilo personale. Una volta selezionato un task, possono procedere con la consegna caricando la propria soluzione, che dovrà essere fornita in formato testuale attraverso un’apposita interfaccia. Il docente ha la facoltà di valutare ogni consegna assegnando un punteggio compreso tra zero e il massimo indicato nel task.

Il sistema mantiene lo storico di tutti i task svolti da ciascuno studente e i relativi punteggi ottenuti.

Ogni studente può visualizzare la classifica relativa esclusivamente alla propria classe, attraverso un’apposita sezione del profilo. Le classifiche disponibili sono due: una basata sul punteggio totale accumulato e una basata sul numero di task completati. Queste classifiche consentono agli studenti di confrontare le proprie performance con quelle degli altri membri della stessa classe, incentivando la partecipazione attiva alle attività didattiche. Per semplicità si può supporre che la classifica mostri solo un numero limitato di studente (ex. i primi tre, o i primi cinque)

Ogni studente ha accesso a un profilo personale dove può consultare il totale dei task assegnati, il punteggio totale ottenuto, e la media dei voti ottenuti. I docenti, attraverso un’interfaccia dedicata, possono visualizzare l’insieme delle classi create e, per ciascuna classe, visualizzare l’elenco degli studenti iscritti e l’elenco dei task assegnati.

Il sistema dovrà essere accessibile via web sia da desktop che da dispositivi mobili, con un’interfaccia grafica intuitiva e responsiva. È previsto un sistema di notifiche che avvisa gli studenti della pubblicazione di nuovi task e dell’avvenuta correzione delle proprie consegne.

**Legenda:**

Classe

Attributi

Funzionalità

Attore

Classe-Attore

**2.2 Revisione dei requisiti**

1. Il sistema deve offrire all’Utente una funzionalità per registrarsi.
2. Il sistema deve verificare che l’Utente non sia già precedentemente registrato.
3. Il sistema deve offrire all’Utente una funzionalità per accedere.
4. Di ogni Utente si vuole memorizzare: ruolo, nome, cognome, e-mail istituzionale.
5. Il sistema deve offrire al Docente una funzionalità per la creazione di classi virtuali.
6. Di ogni classe virtuale si vuole memorizzare un nome e un codice univoco.
7. Il sistema deve offrire al Docente la funzionalità di iscrivere uno Studente ad una classe virtuale.
8. Il sistema deve offrire allo Studente la funzionalità di iscriversi ad una classe.
9. Il sistema deve controllare all’ iscrizione che uno Studente non sia iscritto già in un’altra classe.
10. Il sistema deve offrire al Docente la funzionalità di creare task.
11. Il sistema deve offrire al Docente la funzionalità di assegnare task.
12. Di ogni task si vuole memorizzare: titolo, descrizione, data di scadenza e numero massimo di punti assegnabili.
13. Il sistema controlla che la data di scadenza per una task inserita dal Docente non sia precedente alla data di assegnazione.
14. Il sistema deve offrire agli Studenti una funzionalità per visualizzare l’elenco dei task assegnati.
15. Il sistema deve offrire agli Studenti una funzionalità di consegna della soluzione.
16. Il sistema controlla che il formato sia testuale.
17. Il sistema controlla che la data della consegna della soluzione non superi la data di scadenza.
18. Il sistema deve offrire al Docente una funzionalità per la valutazione dei task.
19. Il sistema controlla che la valutazione non superi il punteggio massimo.
20. Di ogni studente si vuole memorizzare lo storico di tutti i task svolti e i relativi punteggi ottenuti.
21. Il sistema deve offrire agli Studenti una funzionalità per la visualizzazione di una classifica basata sul punteggio totale accumulato relativo esclusivamente alla propria classe.
22. Il sistema deve offrire agli Studenti una funzionalità per la visualizzazione di una classifica basata sul numero di task completati relativa esclusivamente alla propria classe.
23. Il sistema deve offrire allo Studente una funzionalità per accedere al proprio profilo personale.
24. Di ogni profilo personale si vuole memorizzare: totale dei task assegnati, punteggio totale ottenuto e media dei voti ottenuti.
25. Il sistema deve offrire al Docente una funzionalità per la visualizzazione delle classi create.
26. Il sistema deve offrire al Docente la funzionalità, per ciascuna classe, di visualizzare l’elenco degli Studenti iscritti e l’elenco dei task assegnati.
27. Il sistema deve offrire la funzionalità all’Utente di accedere via web, sia da desktop che da dispositivi mobili.
28. Il sistema deve inviare una notifica agli Studenti per l’assegnazione di nuove task e l’avvenuta correzione della consegna.
29. Per verificare la validità dei dati deve essere disponibile un sistema di autenticazione esterno al sistema.
30. Il sistema calcola la media dei voti ottenuti dallo Studente.
31. Il sistema calcola il punteggio totale dei task per ogni Studente.
32. Il sistema mostra il numero di task assegnati per Studente.
33. Il sistema genera due tipi di classifiche per classe virtuale: la prima quando almeno uno studente della classe virtuale consegna la soluzione, la seconda quando il docente valuta almeno una consegna della classe virtuale.

**2.3 Glossario dei termini**

| Termine | Descrizione | Sinonimi |
| --- | --- | --- |
| **Utente** | Studente o Docente che utilizza il sistema | Utilizzatore |
| **Classe Virtuale** | Creata dal Docente, è un insieme di Studenti, che vi si possono iscrivere, o essere iscritti dal Docente | Aula Virtuale |
| **Task** | Assegnazioni del docente che prevedono una valutazione | Compito |
| **Profilo Personale** | Profilo di ogni studente, nella quale è possibile visualizzare: task assegnati, punteggio totale ottenuto di ogni task e la media dei voti | Scheda utente |
| **Soluzione** | Svolgimento del task di uno Studente | Consegna |
| **Interfaccia** | Sistema che interagisce con il docente per la visualizzazione delle classi create e di ogni classe la visualizzazione degli Studenti iscritti e dei relativi task assegnati | Schermata |
| **Notifica** | Messaggio che avvisa lo Studente della pubblicazione di un nuovo task e della sua correzione | Messaggio |

**2.4 Classificazione dei requisiti**

**2.4.1 Requisiti funzionali**

| **ID** | **Requisito** | **Origine** |
| --- | --- | --- |
| **RF01** | Il sistema deve offrire all’Utente una funzionalità per registrarsi | 1 |
| **RF02** | Il sistema deve offrire all’Utente una funzionalità per accedere | 3 |
| **RF03** | Il sistema deve offrire al Docente una funzionalità per la creazione di classi virtuali | 5 |
| **RF04** | Il sistema deve offrire al Docente la funzionalità di iscrivere uno Studente ad una classe virtuale. | 7 |
| **RF05** | Il sistema deve offrire allo Studente la funzionalità di iscriversi ad una classe virtuale | 8 |
| **RF06** | Il sistema deve offrire al Docente la funzionalità di creare task | 10 |
| **RF07** | Il sistema deve offrire al Docente la funzionalità di assegnare task | 11 |
| **RF08** | Il sistema deve offrire agli Studenti una funzionalità per visualizzare l’elenco dei task assegnati | 14 |
| **RF09** | Il sistema deve offrire agli Studenti una funzionalità di consegna della soluzione | 15 |
| **RF10** | Il sistema deve offrire al Docente una funzionalità per la valutazione dei task | 18 |
| **RF11** | Il sistema deve offrire agli Studenti una funzionalità per la visualizzazione di una classifica basata sul punteggio totale accumulato relativo esclusivamente alla propria classe | 21 |
| **RF12** | Il sistema deve offrire agli Studenti una funzionalità per la visualizzazione di una classifica basata sul numero di task completati relativa esclusivamente alla propria classe | 22 |
| **RF13** | Il sistema deve offrire allo Studente una funzionalità per accedere al proprio profilo personale. | 23 |
| **RF14** | Il sistema deve offrire al Docente una funzionalità per la visualizzazione delle classi virtuali create | 25 |
| **RF15** | Il sistema deve offrire al Docente la funzionalità, per ciascuna classe, di visualizzare l’elenco degli Studenti iscritti e l’elenco dei task assegnati | 26 |
| **RF16** | Il sistema deve offrire la funzionalità all’Utente di accedere via web, sia da desktop che da dispositivi mobili | 27 |
| **RF17** | Il sistema deve inviare una notifica agli Studenti per l’assegnazione di nuove task e l’avvenuta correzione della consegna | 28 |
| **RF18** | Il sistema genera due tipi di classifiche per classe virtuale: la prima quando almeno uno Studente della classe virtuale consegna la soluzione, la seconda quando il Docente valuta almeno una consegna della classe virtuale | 33 |

**2.4.2 Requisiti di dati**

| **ID** | **Requisito** | **Origine** |
| --- | --- | --- |
| **RD01** | Di ogni Utente si vuole memorizzare ruolo, nome, cognome, e-mail istituzionale | 4 |
| **RD02** | Di ogni classe virtuale si vuole memorizzare un nome e un codice univoco | 6 |
| **RD03** | Di ogni task si vuole memorizzare titolo, descrizione, data di scadenza e numero massimo di punti assegnabili | 12 |
| **RD04** | Di ogni Studente si vuole memorizzare lo storico di tutti i task svolti e i relativi punteggi ottenuti | 20 |
| **RD05** | Di ogni profilo personale si vuole memorizzare il totale dei task assegnati, punteggio totale ottenuto e media dei voti ottenuti | 24 |

**2.4.3 Vincoli/Altri requisiti**

| **ID** | **Requisito** | **Origine** |
| --- | --- | --- |
| **V01** | Il sistema deve verificare che l’Utente non sia già precedentemente registrato | 2 |
| **V02** | Il sistema deve controllare, all’ iscrizione, che uno Studente non sia iscritto già in un’altra classe virtuale | 9 |
| **V03** | Il sistema controlla che la data di scadenza per una task inserita dal Docente non sia precedente alla data di assegnazione | 13 |
| **V04** | Il sistema controlla che il formato della soluzione caricata dagli Studenti sia testuale | 16 |
| **V05** | Il sistema controlla che la data della consegna della soluzione non superi la data di scadenza | 17 |
| **V06** | Il sistema controlla che la valutazione di un task non superi il punteggio massimo | 19 |
| **V07** | Per verificare la validità dei dati deve essere disponibile un sistema di autenticazione esterno al sistema | 29 |
| **V08** | Il sistema calcola la media dei voti ottenuti dallo Studente | 30 |
| **V09** | Il sistema calcola il punteggio totale dei task per ogni Studente | 31 |
| **V10** | Il sistema mostra il numero di task assegnati | 32 |

**2.5 Modellazione dei casi d’uso**

**2.5.1 Attori e casi d’uso**

**Attori Primari:** **Attori Secondari:**

- Studenti -Servizio di Messaggistica

- Docenti

-Utente

-Utente non registrato

**Casi d’uso:** **Casi d’uso di inclusione:**

**UC1** Registrazione **UC13** Verifica credenziali

**UC2** Accesso **UC14** Verifica scadenza task

**UC3** Creazione Classe Virtuale **UC15** Accesso profilo personale

**UC4** Iscrizione classe Docente **UC16** Invio messaggio

**UC5** Iscrizione classe Studente

**UC6** Creazione task

**UC7** Assegnazione task

**UC8** Carica soluzione

**UC9** Valutazione consegna

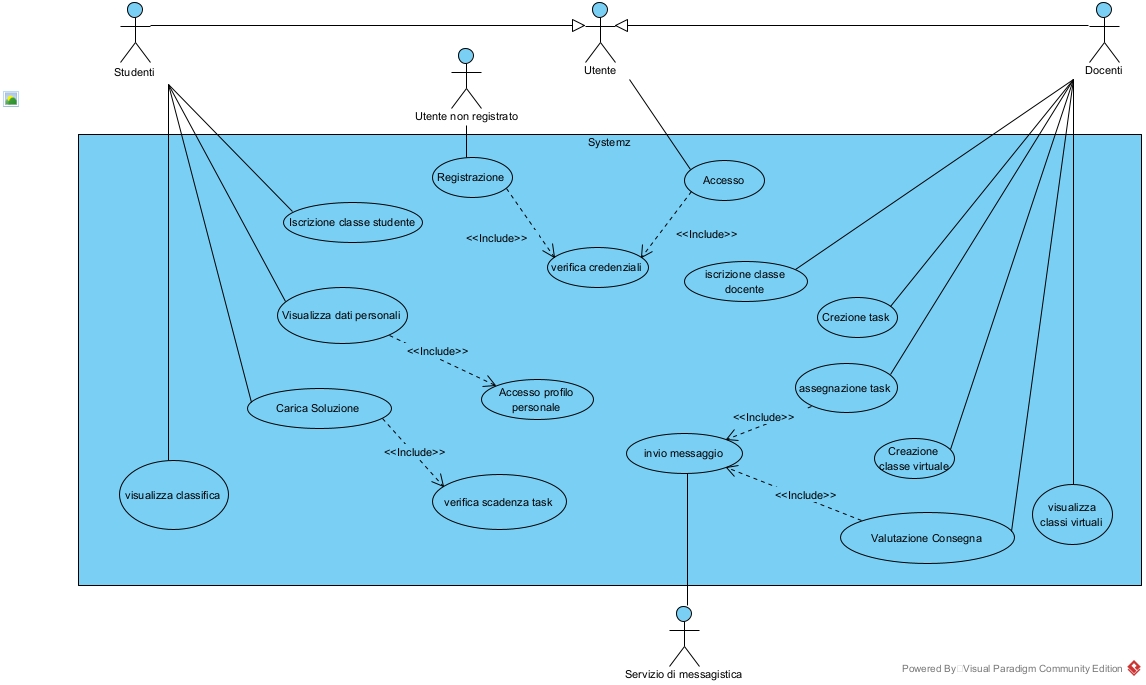
**UC10** Visualizza classifica

**UC11** Visualizza dati personali

**UC12** Visualizza classi virtuali

| **Caso d’uso** | **Attori Primari** | **Attori Secondari** | **Include/ Extends** | **Requisiti corrispondenti** |
| --- | --- | --- | --- | --- |
| **UC1:**Registrazione | Utente non registrato |  | ***Include***  verifica delle credenziali | RF01 |
| **UC2:**Accesso | Utente |  | ***Include***  verifica delle credenziali | RF02 |
| **UC3:**Creazione Classe Virtuale | Docente |  |  | RF03 |
| **UC4:**Iscrizione classe  Docente | Docente | Studente |  | RF04 |
| **UC5:**Iscrizione classe  Studente | Studente |  |  | RF05 |
| **UC6:**Creazione task | Docente |  |  | RF06 |
| **UC7:**Assegnazione task | Docente | Servizio di messaggistica | ***Include***  invio messaggio | RF07 |
| **UC8:** Carica soluzione | Studente |  | ***Include***  verifica scadenza task  ***Extends***  genera classifica per numero di task svolti | RF09 |
| **UC9:**Valutazione consegna | Docente | Servizio di messaggistica | ***Include***  invio messaggio  ***Extends***  genera classifica per punteggio totale accumulato | RF10 |
| **UC10:**Visualizza classifica | Studente |  |  | RF11  RF12 |
| **UC11:**Visualizza dati personali | Studente |  | ***Include***  accesso profilo personale | RF13 |
| **UC12:** Visualizza classi virtuali | Docente |  |  | RF14 |

**2.5.2 Diagramma dei casi d’uso**



**2.5.3 Scenari**

| Caso d’uso: | Registrazione |
| --- | --- |
| Attore primario | Utente non registrato |
| Attore secondario | - |
| Descrizione | L’utente si registra alla piattaforma |
| Pre-Condizioni | Le credenziali devono essere diverse per ogni utente |
| Sequenza di eventi  principale | 1. Il caso d’uso inizia quando l’utente si connette alla piattaforma per la registrazione. 2. L’utente immette le credenziali per la registrazione. 3. . <<include>> *VerificaCredenziali.* 4. **if** credenziali non presenti nel sistema.   4.1 Il sistema salva le informazioni dell’utente nella piattaforma   1. Il caso d’uso termina. |
| Post-Condizioni | - |
| Casi d’uso correlati | *Verifica credenziali* |
| Sequenza di eventi  alternativi | Al punto 4 se nel sistema è presente un’altro utente con la stessa e-mail istituzionale il sistema restituisce un messaggio di errore, mettendo a disposizione la possibilità di reinserire le credenziali |

| Caso d’uso: | Accesso |
| --- | --- |
| Attore primario | Utente |
| Attore secondario | - |
| Descrizione | L’utente accede alla piattaforma |
| Pre-Condizioni | Le credenziali devono essere presenti nel sistema |
| Sequenza di eventi  principale | 1. Il caso d’uso inizia quando l’utente si connette alla piattaforma per l’accesso. 2. L’utente immette le credenziali d’accesso. 3. <<include>> *VerificaCredenziali.* 4. **if** le credenziali sono presenti nel sistema   4.1 L’utente accede alla piattaforma.   1. Il caso d’uso termina. |
| Post-Condizioni | - |
| Casi d’uso correlati | *Verifica credenziali* |
| Sequenza di eventi  alternativi | Al punto 2 se nel sistema non sono presenti le credenziali immesse dall’utente il sistema genera un messaggio di errore e permette di reinserire le credenziali |

| Caso d’uso: | Creazione classe virtuale |
| --- | --- |
| Attore primario | Docente |
| Attore secondario | - |
| Descrizione | Il docente crea una classe virtuale |
| Pre-Condizioni | - |
| Sequenza di eventi  principale | 1. Il caso d’uso inizia quando il docente accede alla sezione dedicata alla creazione delle classi virtuali. 2. Il docente inserisce inserisce le credenziali della classe virtuale.   2.1 Il sistema controlla che il codice inserito non sia già presente.   1. Il docente conferma la creazione della classe. 2. Il caso d’uso termina. |
| Post-Condizioni | - |
| Casi d’uso correlati | ***-*** |
| Sequenza di eventi  alternativi | Al punto 2.1 se il codice univoco è già presente nel sistema, questo genera un messaggio di errore |

| Caso d’uso: | Iscrizione classe Docente |
| --- | --- |
| Attore primario | Docente |
| Attore secondario | Studente |
| Descrizione | Il docente iscrive lo studente ad una classe virtuale |
| Pre-Condizioni | Il docente deve aver creato almeno una classe virtuale |
| Sequenza di eventi  principale | 1. Il caso d’uso inizia quando il docente accede alla sezione dedicata alle iscrizioni degli studenti alle classi virtuali. 2. Il docente seleziona la classe virtuale in cui iscrivere lo studente. 3. Il docente ricerca lo studente da iscrivere alla classe tramite la sua e-mail.   3.1 Il sistema controlla se la mail appartiene o meno ad uno studente.  3.2 Il sistema controlla se lo studente è già iscritto ad una classe virtuale.   1. Il docente conferma l’iscrizione. 2. Il caso d’uso termina. |
| Post-Condizioni | - |
| Casi d’uso correlati | - |
| Sequenza di eventi  alternativi | Al punto 3.1 se la mail non appartiene a nessuno studente il sistema genera un messaggio di errore. Al punto 3.2 se uno degli studenti risulta già iscritto ad una classe, il sistema genera un messaggio d’errore. |

| Caso d’uso: | Iscrizione classe studente |
| --- | --- |
| Attore primario | Studente |
| Attore secondario | - |
| Descrizione | Lo studente si iscrive alla classe virtuale |
| Pre-Condizioni | - |
| Sequenza di eventi  principale | 1. Il caso d’uso inizia quando lo studente accede alla sezione per l’iscrizione ad una classe virtuale. 2. Lo studente inserisce il codice univoco della classe virtuale.   2.1 Il sistema controlla che il codice univoco esista.   1. Il sistema iscrive lo studente alla classe virtuale. 2. Il caso d’uso termina. |
| Post-Condizioni | - |
| Casi d’uso correlati | ***-*** |
| Sequenza di eventi  alternativi | Al punto 2.1 se il codice inserito è inesistente il sistema genera un messaggio di errore. |

| Caso d’uso: | Creazione task |
| --- | --- |
| Attore primario | Docente. |
| Attore secondario | - |
| Descrizione | Il docente crea un task. |
| Pre-Condizioni | Il docente deve appartenere a quella classe. |
| Sequenza di eventi  principale | 1. Il caso d’uso inizia quando il docente accede alla sezione dedicata alla creazione dei task. 2. Il docente inserisce le specifiche richieste. 3. **if** titolo non trovato 4. **if** data scadenza>data odierna   4.1 il sistema da la possibilità di confermare la creazione del task   1. Il docente conferma la creazione del task. 2. Il sistema salva il task creato nell’ apposita lista delle classi virtuali 3. Il caso d’uso termina. |
| Post-Condizioni | Il task appare nell’elenco dei task assegnabili |
| Casi d’uso correlati | *-* |
| Sequenza di eventi  alternativi | Al punto 3 se il titolo già esiste nel sistema, questo invia un messaggio di errore.  Al punto 4 se la data di scadenza non è corretta il sistema invia un messaggio di errore. |

| Caso d’uso: | Assegnazione task |
| --- | --- |
| Attore primario | Docente |
| Attore secondario | Servizio di messaggistica |
| Descrizione | Il docente assegna un task ad una classe virtuale, il servizio di messaggistica informa gli studenti della classe virtuale dell’avvenuta consegna. |
| Pre-Condizioni | Il task deve essere già stato creato. |
| Sequenza di eventi  principale | 1. Il caso d’uso inizia quando il docente accede alla sezione dedicata all’assegnazione task. 2. Il docente seleziona la classe virtuale a cui assegnare il task. 3. Il docente seleziona il task da assegnare. 4. Il docente conferma l’assegnazione. 5. <<include>> *InvioMessaggio.* 6. Il caso d’uso termina. |
| Post-Condizioni | Il task compare nell’elenco task degli studenti interessati |
| Casi d’uso correlati | *Invio messaggio* |
| Sequenza di eventi  alternativi | Al punto 4 il sistema invia un messaggio d’errore se il task selezionato è stato precedentemente assegnato alla classe virtuale. |

| Caso d’uso: | Carica soluzione |
| --- | --- |
| Attore primario | Studente |
| Attore secondario |  |
| Descrizione | Lo studente carica la soluzione in formato testuale nell’apposita sezione |
| Pre-Condizioni | Lo studente deve selezionare il task a cui assegnare la soluzione |
| Sequenza di eventi  principale | 1. Il caso d’uso inizia quando lo studente seleziona il task da svolgere. 2. Lo studente scrive la soluzione. 3. Lo studente conferma la soluzione. 4. <<include>> *VerificaScadenzaTask.* 5. **if** la consegna rispetta la scadenza   5.1 la consegna viene contrassegnata come “consegnata”   1. Il sistema aggiunge la soluzione alla lista delle soluzioni del task interessato. 2. Il caso d’uso termina. |
| Post-Condizioni | Il task viene contrassegnato come “consegnato” e inserito nella lista dei task completati dallo studente. |
| Casi d’uso correlati | *Verifica scadenza task* |
| Sequenza di eventi  alternativi | Al punto 5.1 se la soluzione del task viene consegnata in ritardo il task viene contrassegnato come “consegnato in ritardo”. |

| Caso d’uso: | Valutazione consegna |
| --- | --- |
| Attore primario | Docente |
| Attore secondario | Servizio messaggistica |
| Descrizione | Il docente assegna un punteggio alla soluzione caricata dallo studente |
| Pre-Condizioni | La soluzione deve essere presente |
| Sequenza di eventi  principale | 1. Il caso d’uso inizia quando il docente accede all’interfaccia dedicata alla valutazione dei task. 2. Il sistema mostra al docente i task da valutare. 3. Il sistema mostra al docente le soluzioni. 4. Il docente seleziona la soluzione da valutare. 5. Il docente assegna un punteggio alla consegna.   5.1 Il sistema controlla che il punteggio assegnato non sia superiore a quello prefissato.   1. Il docente conferma la valutazione. 2. Il sistema salva la valutazione. 3. <<include>> *InvioMessaggio*. 4. Il caso d’uso termina. |
| Post-Condizioni | Il task viene contrassegnato come "valutato". |
| Casi d’uso correlati | *Invio messaggio* |
| Sequenza di eventi  alternativi | Nel punto 5.1 se il punteggio assegnato è superiore a quello prefissato il sistema genera un messaggio di errore. |

| Caso d’uso: | Visualizza classifica |
| --- | --- |
| Attore primario | Studente |
| Attore secondario |  |
| Descrizione | Lo studente visualizza una delle due classifiche disponibili |
| Pre-Condizioni | Deve esistere almeno una classifica |
| Sequenza di eventi  principale | 1. Il caso d’uso inizia quando lo studente accede alla sezione dedicata alla visualizzazione delle classifiche 2. Lo studente sceglie quale delle due classifiche visionare. 3. Il sistema mostrerà i primi 5 studenti della classe che rispecchiano i criteri della classifica. 4. Il caso d’uso termina quando lo studente esce dalla sezione dedicata alla visualizzazione delle classifiche. 5. Il caso d’uso termina. |
| Post-Condizioni | - |
| Casi d’uso correlati | ***-*** |
| Sequenza di eventi  alternativi | Al punto 2 se una o entrambe le classifiche non sono state generate il sistema non mette a disposizione la selezione allo studente. |

| Caso d’uso: | Visualizza dati personali |
| --- | --- |
| Attore primario | Studente |
| Attore secondario |  |
| Descrizione | Lo studente visualizza i dati personali |
| Pre-Condizioni | - |
| Sequenza di eventi  principale | 1. <<include>> *AccessoProfiloPersonale*. 2. Il sistema mostrerà i dati dello studente e i dati relativi ai task che lo interessano. 3. Il caso d’uso termina quando lo studente esce dal profilo personale. 4. Il caso d’uso termina. |
| Post-Condizioni | - |
| Casi d’uso correlati | *Accesso profilo personale* |
| Sequenza di eventi  alternativi | - |

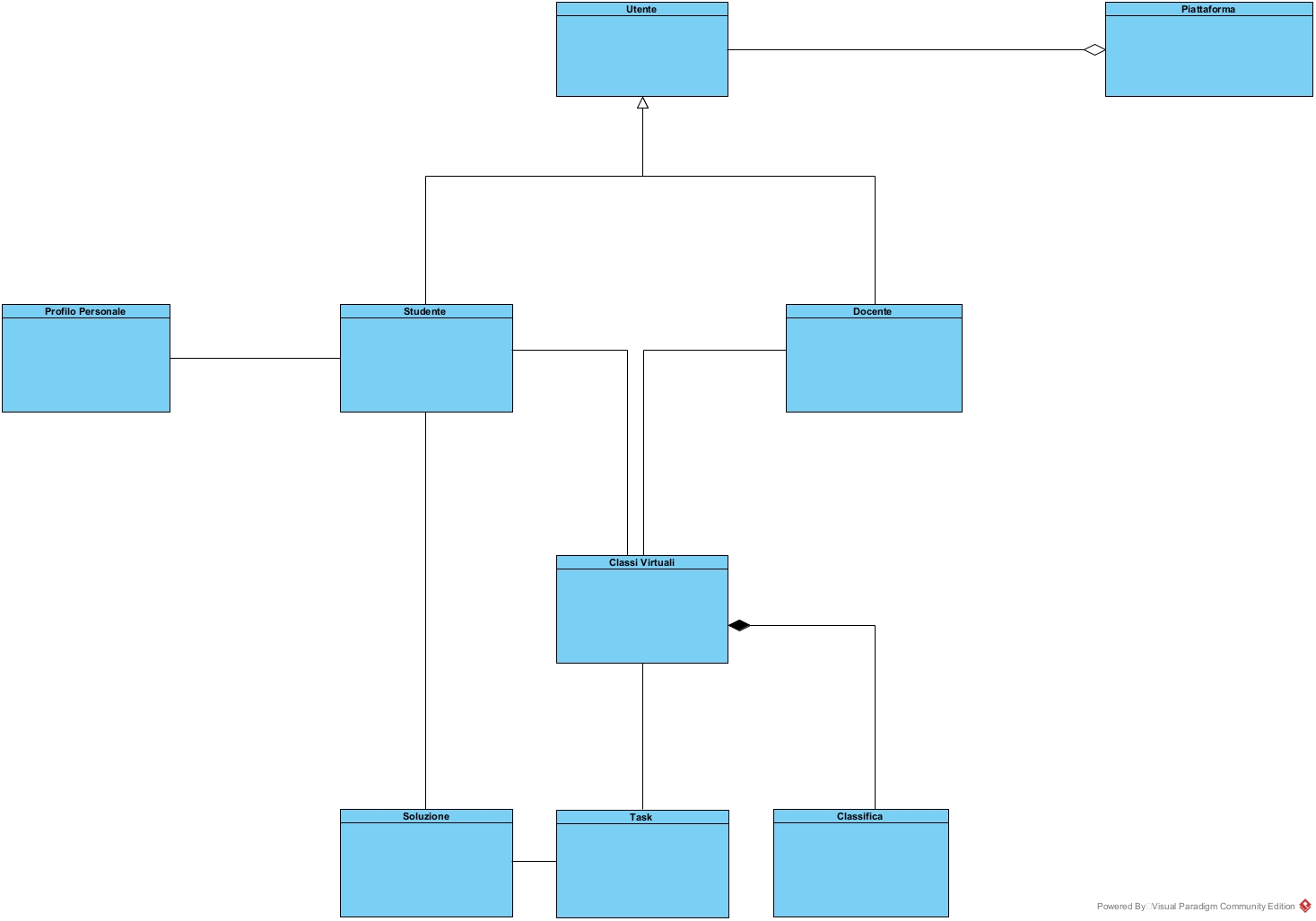
## 

| Caso d’uso: | Visualizza classi virtuali |
| --- | --- |
| Attore primario | Docente |
| Attore secondario | - |
| Descrizione | Il docente visualizza l'elenco delle sue classi virtuali e i relativi studenti e task appartenenti alle classi |
| Pre-Condizioni | Deve esistere almeno una classe virtuale |
| Sequenza di eventi  principale | 1. Il caso d’uso inizia quando il docente accede alla sezione relativa alla visualizzazione delle classi virtuali. 2. Il sistema mostra l’elenco delle classi virtuali del docente e i relativi studenti e task. 3. Il caso d’uso termina. |
| Post-Condizioni | - |
| Casi d’uso correlati | ***-*** |
| Sequenza di eventi  alternativi | - |

## 

**2.6 Diagramma delle classi**

**Diagramma base**

****

**2.6.1 Responsabilità**

| **RESPONSABILITÀ** | **CLASSE** |
| --- | --- |
| Registrazione | Sistema |
| Accesso | Sistema |
| Creazione Classe Virtuale | Docente |
| Iscrizione Classe Docente | Docente |
| Iscrizione Classe Studente | Studente |
| Creazione Task | Docente |
| Assegnazione Task | Docente |
| Carica Soluzione | Studente |
| Valutazione Consegna | Docente |
| Visualizza Classifica | Studente |
| Visualizza Dati Personali | Studente |
| Visualizza Classi VIrtuali | Docente |
| Verifica Credenziali | Sistema |
| Verifica Scadenza Task | Data |
| Accesso Profilo Personale | Studente |
| Invio Messaggio | Sistema |

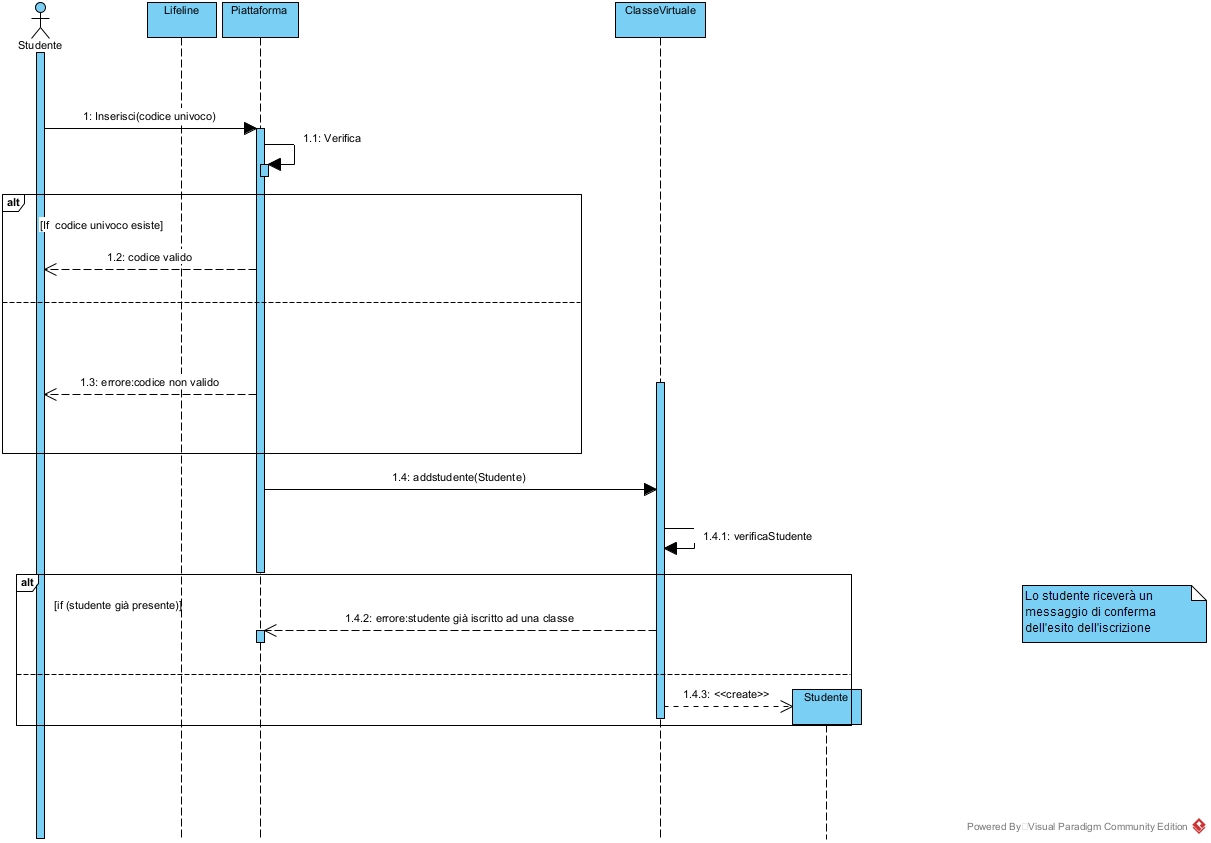
**Descrizione delle assegnazioni di responsabilità con i Pattern GRASP**

* **Registrazione** e **Accesso** sono responsabilità di **Sistema**, in quanto <<Information Expert>> di **Utente.**
* **Creazione Classe Virtuale** è responsabilità di **Docente**, in quanto <<Creator>> di **Classe Virtuale.**
* **Iscrizione Classe Docente** è responsabilità di **Docente**, in quanto <<Low Coupling>> di **Classe Virtuale.**
* **Iscrizione Classe Studente** è responsabilità di **Studente**, in quanto <<Low Coupling>> di **Classe Virtuale.**
* **Crea Task** è responsabilità di **Docente**, in quanto <<Creator>> di **Task.**
* **Assegnazione Task** è responsabilità di **Docente**, in quanto <Low Coupling> di **Classe Virtuale.**
* **Carica Soluzione** è responsabilità di **Studente**, in quanto <<Creator>> di **Soluzione.**
* **Valutazione Consegna** è responsabilità di **Docente**, in quanto <<Information expert>> di **Task** e **Soluzione.**
* **Visualizza Classifica** è responsabilità di **Studente**, in quanto <<Low Coupling>> di **Classe Virtuale.**
* **Visualizza dati personali** è responsabilità di **Studente**, in quanto <<Low Coupling>> della classe **Profilo Personale.**
* **Visualizza Classi Virtuali** è responsabilità del **Docente**, in quanto <<Low Coupling>> di **Classi Virtuali.**

**2.7 Diagrammi di Sequenza**

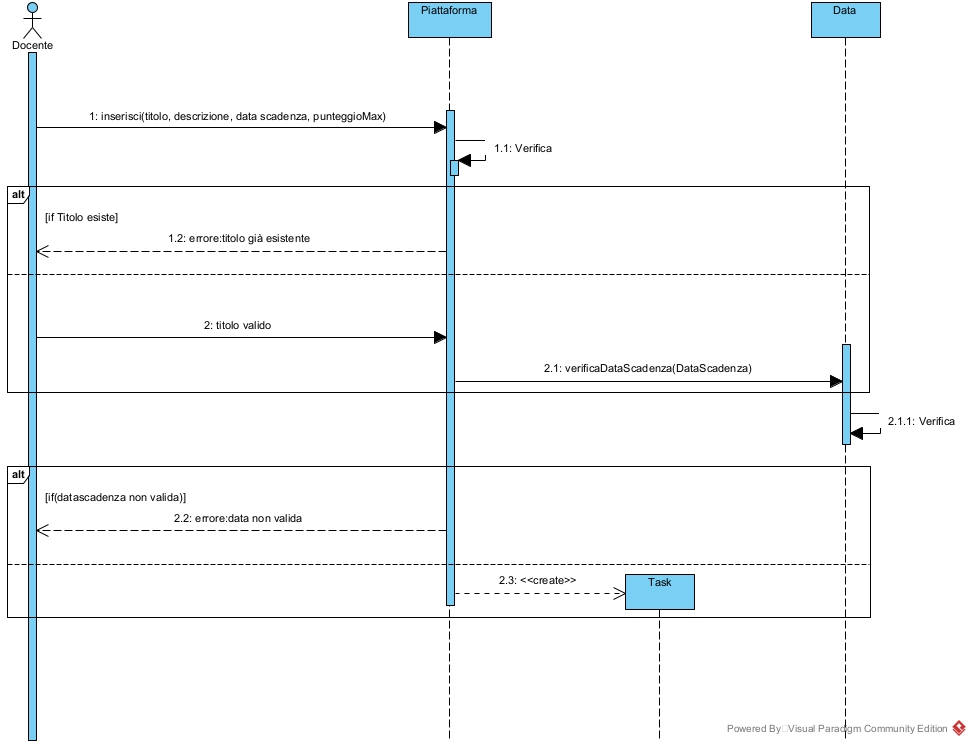
**2.7.1 Iscrizione Classe Studente**

Nel seguente diagramma è stato implementato un metodo “searchClasseVirtuale”, il quale cerca la classe in cui si vuole registrare lo studente tramite il codice univoco della classe interessata. Inoltre il sistema verifica che lo studente non sia già iscritto ad una classe virtuale.



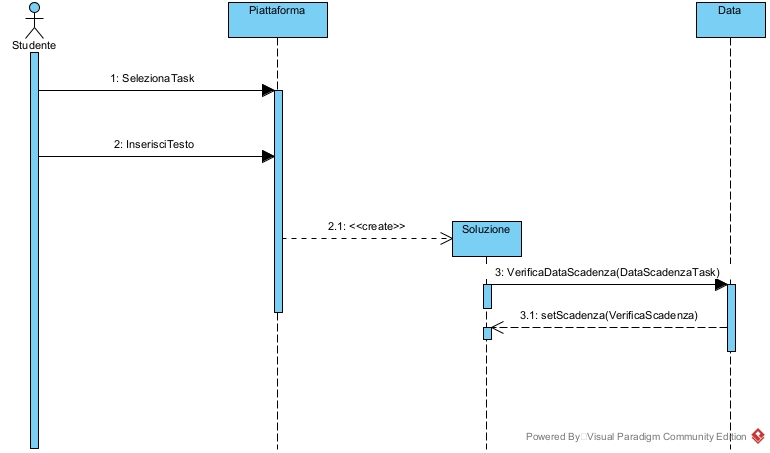
**2.7.2 Creazione Task**

Nel seguente diagramma è stato necessario inserire due controlli: uno per verificare se il task esiste già, cercando nella lista dei task creati il titolo, il secondo per verificare che la data di scadenza inserita dal docente non sia inferiore o uguale al giorno di assegnazione del task.

****

**2.7.3 Carica Soluzione**

Nel seguente sequence diagram è risultato necessario inserire un metodo *verifica scadenza soluzione* che confronta la data di scadenza del task precedentemente assegnato dal docente con la data di consegna della soluzione. In caso di consegna rispettata la soluzione viene contrassegnata come “Consegnata” e il metodo *setScadenza* modifica il valore del booleano “Consegnato” in soluzione come “True”, in caso contrario la soluzione viene contrassegnata come “Consegnata in ritardo” e il metodo modifica “Consegnato” in “False”.



**Diagramma raffinato**

****

**3. Piano di test funzionale**

**3.1 Creazione Task**

| Creazione task | | | |
| --- | --- | --- | --- |
| **Titolo** | **Descrizione** | **Data Scadenza** | **Punteggio Massimo** |
| * Stringa non vuota di lunghezza ≤ 50 * Titolo già presente nel sistema **[ERROR]** * Stringa vuota o troppo lunga **[ERROR]** * Stringa con caratteri speciali non ammessi **[ERROR]** | * Testo descrittivo ≤ 1000 caratteri * Stringa vuota o > 1000 caratteri **[ERROR]** | ▪ Data inserita > Data corrente  ▪ Data inserita < Data corrente  **[ERROR]**  ▪ Formato data non valido **[ERROR]** | ▪ Numero intero positivo  ▪ Numero negativo **[ERROR]**  ▪ Valore non numerico **[ERROR]** |

Con i vincoli [ERROR], invece, il numero di test da eseguire per testare singolarmente i vincoli è 8 (2 per Titolo, 1 per Descrizione, 2 per Data Scadenza, 2 per Punteggio max).

Il numero di test risultante è: (1\*1\*1\*1) + 7 = 8.

| **TEST SUITE** |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test case ID | Descrizione | Classi di equivalenza coperte | Pre-condizioni | Input | Output attesi | Post-condizioni attese |
| 1 | Tutti input validi | Titolo valido, Descrizione valida, Data Scadenza valida, Punteggio max valido | // | {Titolo: “Compito di italiano", Descrizione:”Tema su vita e opere di Giovanni Pascoli”, Data di scadenza:”01-02-2027”,Punteggio max: “10”} | Task creato | Il task compare nella lista dei task creati della classe virtuale |
| 2 | Titolo stringa >50 caratteri | Titolo stringa>50 [ERROR], Descrizione,Data Scadenza, Punteggio max validi | // | {**Titolo:** “aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa”,Descrizione:”Tema su vita e opere di Giovanni Pascoli”, Data di scadenza:”01-02-2027”,Punteggio max: “10”} | Titolo non valido | // |
| 3 | Titolo stringa con caratteri speciali | Titolo stringa con simboli [ERROR],Descrizione,Data Scadenza, Punteggio max validi | // | {**Titolo**: “@#Compito”,Descrizione:”Tema su vita e opere di Giovanni Pascoli”, Data di scadenza:”01-02-2027”,Punteggio max: “10”} | Titolo non valido | // |
| 4 | Titolo già presente nel sistema | Titolo già presente nel sistema [ERROR] ,Descrizione,Data Scadenza, Punteggio max validi |  | {Titolo: “Compito di italiano", Descrizione:”Tema su vita e opere di Giovanni Pascoli”, Data di scadenza:”01-02-2027”,Punteggio max: “10”} | Titolo già esistente |  |
| 5 | Descrizione stringa vuota o maggiore di 1000 caratteri | Titolo valido, Descrizione stringa vuota o >1000 [ERROR], Data Scadenza, Punteggio max validi | // | {Titolo: “Compito di Matematica”,**Descrizione**:” ”, Data di scadenza:”01-02-2027”,Punteggio max: “10”} | Descrizione troppo lunga o inesistente | // |
| 6 | Data di scadenza inserita già passata | Titolo valido, Descrizione valida, Data di Scadenza passata [ERROR], Punteggio max valido | // | {Titolo: “Compito di Matematica”,Descrizione:”2+2= ”, **Data di scadenza:**”01-02-2007”,Punteggio max: “10”} | Data di scadenza inserita già passata | // |
| 7 | Formato data non valido |  |  | {Titolo: “Compito di Matematica”,Descrizione:”2+2= ”, **Data di scadenza:**”ABCD”,Punteggio max: “10”} |  |  |
| 8 | Punteggio max numero negativo | Titolo valido, Descrizione valida, Data scadenza valida, Punteggio max numero negativo [ERROR]. | // | {Titolo: “Compito di Matematica”,Descrizione:”2+2= ”, Data di scadenza**:**”01-02-2027”,**Punteggio max**: “-10”} | Punteggio max non valido | // |
| 9 | Punteggio max valore non numerico | Titolo valido, Descrizione valida, Data scadenza valida, Punteggio max valore non numerico [ERROR]. |  | {Titolo: “Compito di Matematica”,Descrizione:”2+2= ”, Data di scadenza**:**”01-02-2027”,**Punteggio max**: “ABC”} | Punteggio max non valido | // |

**3.2 Carica Soluzione**

| Carica Soluzione | |
| --- | --- |
| **Testo** | **Data soluzione** |
| * Testo non vuoto o di lunghezza <= 5000 caratteri * Testo vuoto o di lunghezza >5000 caratteri **[ERROR]** | * Consegna avvenuta nei tempi previsti dalla data di scadenza del task * Consegna avvenuta in ritardo rispetto alla data di scadenza del task |

| **TEST**  **SUITE** | | | | | | |
| --- | --- | --- | --- | --- | --- | --- |
| Test case ID | Descrizione | Classi di equivalenza coperte | Pre-condizioni | Input | Output attesi | Post-condizioni attese |
| 1 | Tutti input validi | Testo valido | // | {**Testo**:“La soluzione al quesito 1 è 5.  La soluzione al quesito 2 è 67”} | Soluzione aggiunta | La soluzione viene aggiunta alla lista delle soluzioni del task |
| 2 | Stringa testo vuota o troppo lungo | Testo vuoto o di lunghezza >5000 caratteri [ERROR] | // | {**Testo**:“ ”} | Aggiungi testo o 5000 caratteri superati |  |
| 3 | Consegna effettuata oltre la data di scadenza | Testo valido, data oltre il termine | // | {Testo: “La soluzione al quesito 1 è 5”, **Data** **consegna**: 2025-06-13}, deadline = 2025-06-12 | Consegna” in ritardo“Soluzione aggiunta” | La soluzione viene aggiunta alla lista delle soluzioni del task con flag “in ritardo” |

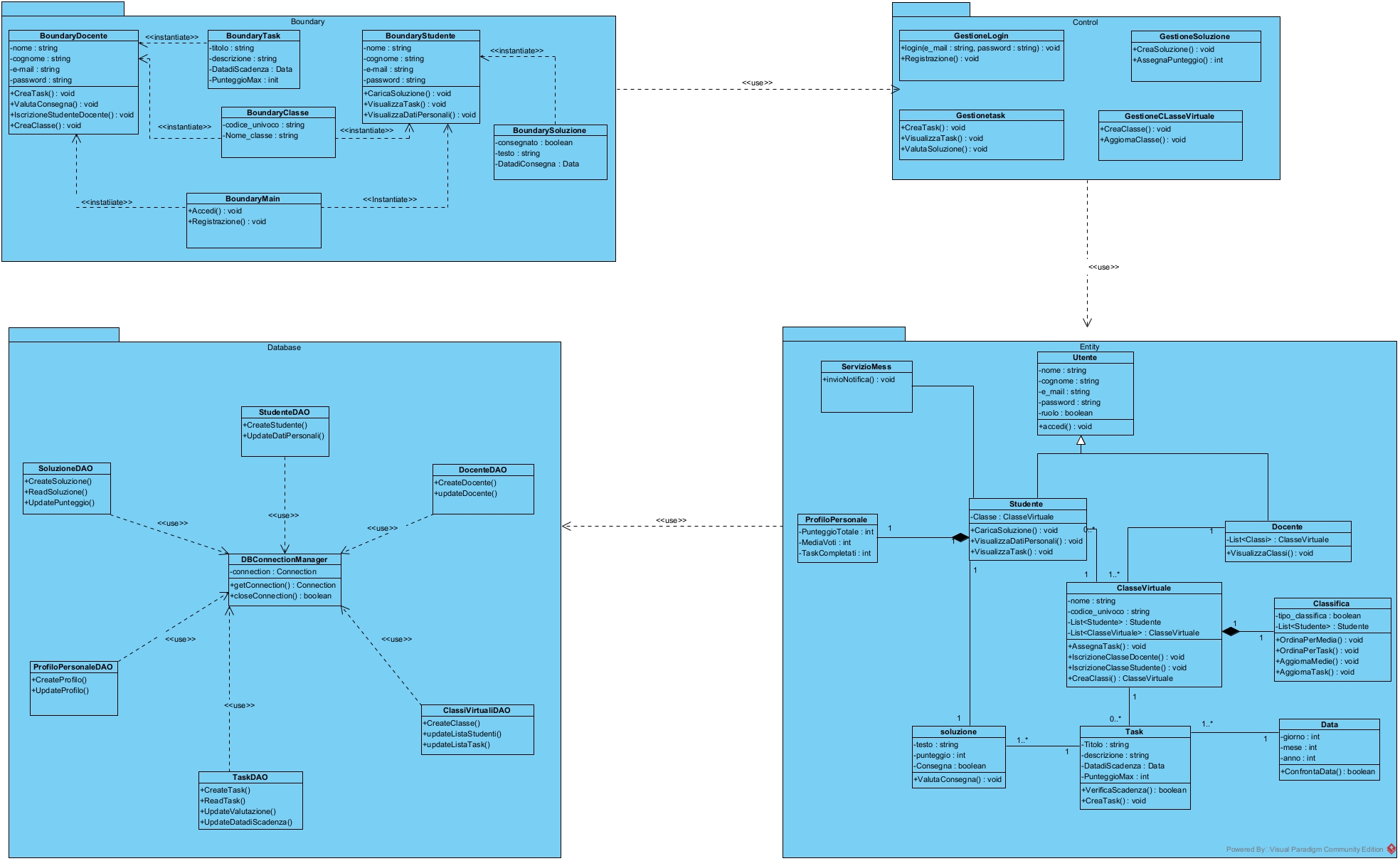
**3.3 Iscrizione Classe Studente**

| ISCRIZIONE CLASSE STUDENTE | |
| --- | --- |
| **Codice Univoco** | **Iscrizione classe** |
| * Stringa non vuota o di lunghezza <=5 * Stringa vuota, >5 o con caratteri speciali **[ERROR]** | * Lo studente non è iscritto ad alcuna classe virtuale * Lo studente è già iscritto ad una classe virtuale **[ERROR]** |

| **TEST**  **SUITE** | | | | | | |
| --- | --- | --- | --- | --- | --- | --- |
| Test case ID | Descrizione | Classi di equivalenza coperte | Pre-condizioni | Input | Output attesi | Post-condizioni attese |
| 1 | Tutti input validi | Codice valido | // | {**Testo**:"ABC 23"} | Aggiunto alla classe | Lo studente viene aggiunto alla classe associata al codice |
| 2 | Stringa testo vuota, troppo lungo o con caratteri speciali | Codice: Stringa vuota o > 5 o con caratteri speciali [ERROR] | // | {**Testo**:“ ”} | Aggiungi codice o codice errato |  |
| 3 | Lo Studente è già iscritto ad una classe virtuale | Codice valido | Lo studente è iscritto ad una classe virtuale | {Testo:"ABC 23"} | Sei già iscritto ad una classe | Lo studente rimane iscritto alla classe precedente |

**4. Progettazione**

**4.1 Diagramma delle classi**

****

**4.1.1 Traduzione classi ed associazioni**

Nel diagramma entity:

* Docente e Studente rappresentano gli Utenti.Ogni studente è associato 1:1 con un Profilo personale mentre, ogni Docente è associato 1:N con le classi virtuali.
* Classe Virtuale è gestita da Docente è contiene un associazione 1:N con i Task.
* Soluzione rappresenta la consegna dello Studente riferita al Task assegnato dal Docente. L’associazione Soluzione-Studente è 1:1, l’associazione fra Soluzione e Task N:1

Nel diagramma controller:

* GestioneLogin gestisce accesso e registrazione.
* GestioneNotifica si occupa della creazione del messaggio di conferma da inviare ad avvenuta valutazione o assegnazione task.
* GestioneTask gestisce la creazione e assegnazione dei task.
* GestioneClassifica si occupa della creazione e dell’aggiornamento delle due classifiche.

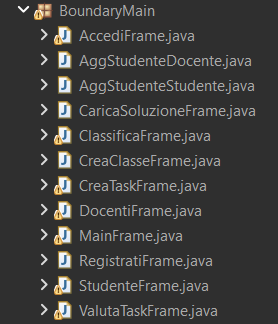
Nel diagramma Boundary:

* Utente non registrato: può solo registrarsi.
* Utente: accede al sistema dopo il login (può essere docente o studente).
* Docente: crea classi, assegna task, valuta soluzioni.
* Studente: svolge i task, riceve notifiche, visualizza profilo e classifica.
* Servizio di messaggistica: invia notifiche agli studenti (es. nuovo task, correzione ricevuta)

Le classi del package Database si occupano della **persistenza dei dati**..

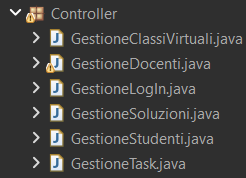
**4.1.2 Pattern BCED**

**4.1.2.1 Package Boundary**

****

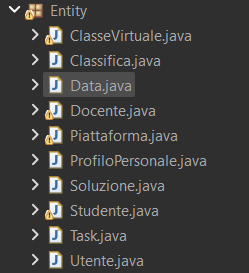
Il package *boundary* include le interfacce che consentono all’utente di inserire i dati necessari per utilizzare le funzionalità offerte dal layer *Control*. Questi dati vengono poi inviati al layer *Control* che, seguendo l’architettura a livelli, elabora le informazioni e restituisce il risultato al *boundary*, il quale si occupa di mostrarlo all’utente.

**4.1.2.2 Package Controller**



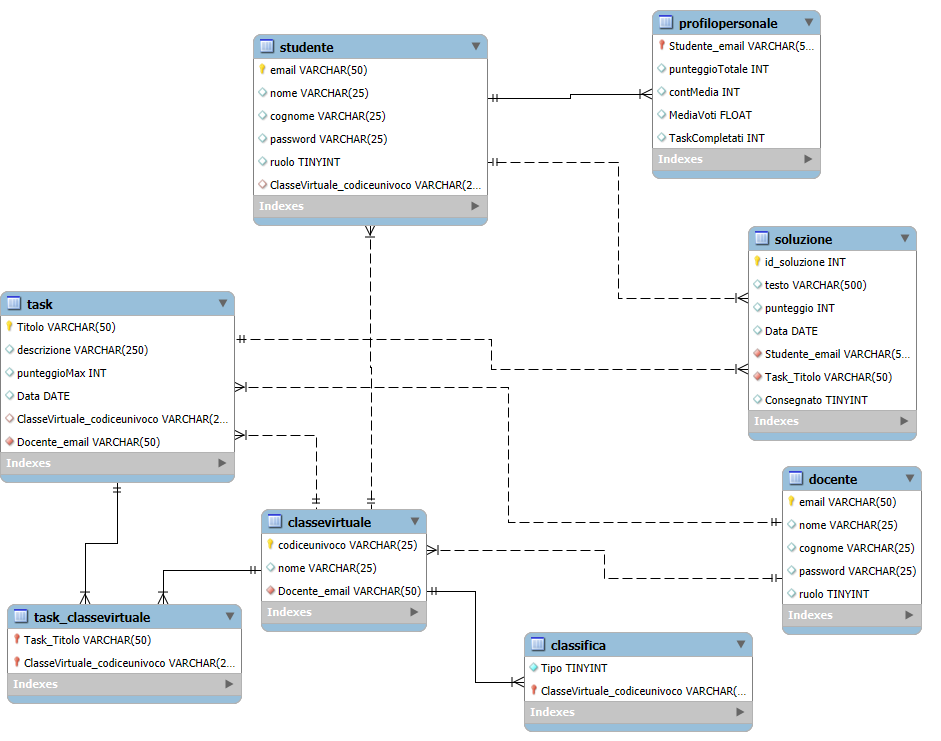
Il package *control* gestisce la logica e il flusso dell’applicazione. Riceve i dati dal *boundary*, elabora le richieste usando le classi *entity* e *DAO*, e restituisce i risultati al *boundary* per mostrarli all’utente.

**4.1.2.3 Package Entity**

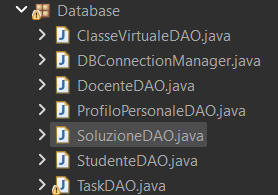


Il package *entity* contiene le classi che rappresentano i dati principali del sistema, come utenti, task o classi virtuali. Queste classi modellano le entità reali con i loro attributi e metodi, fungendo da “mattoncini” su cui si basa tutta l’applicazione.

**4.1.2.4 Package Database**

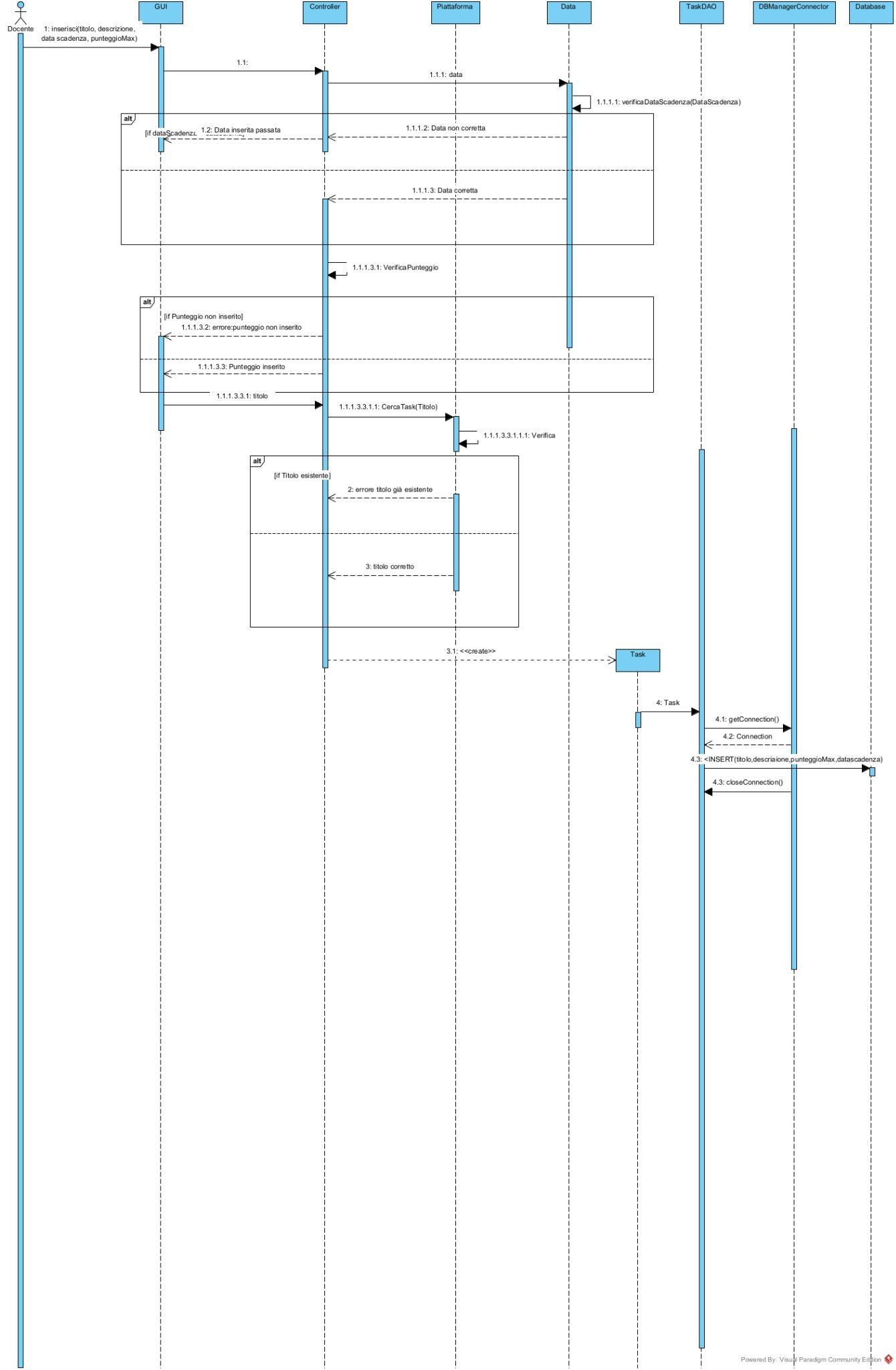


Il package *database* si occupa di gestire la comunicazione tra l’applicazione e il database vero e proprio. Contiene le classi che eseguono operazioni come salvare, aggiornare, eliminare e leggere i dati nel database, usando query e connessioni. In pratica, fa da ponte tra il mondo software e il sistema di archiviazione dei dati.Il *DBManager* gestisce la connessione al database, assicurando che l’applicazione possa comunicare in modo sicuro ed efficiente con il sistema di archiviazione dati.

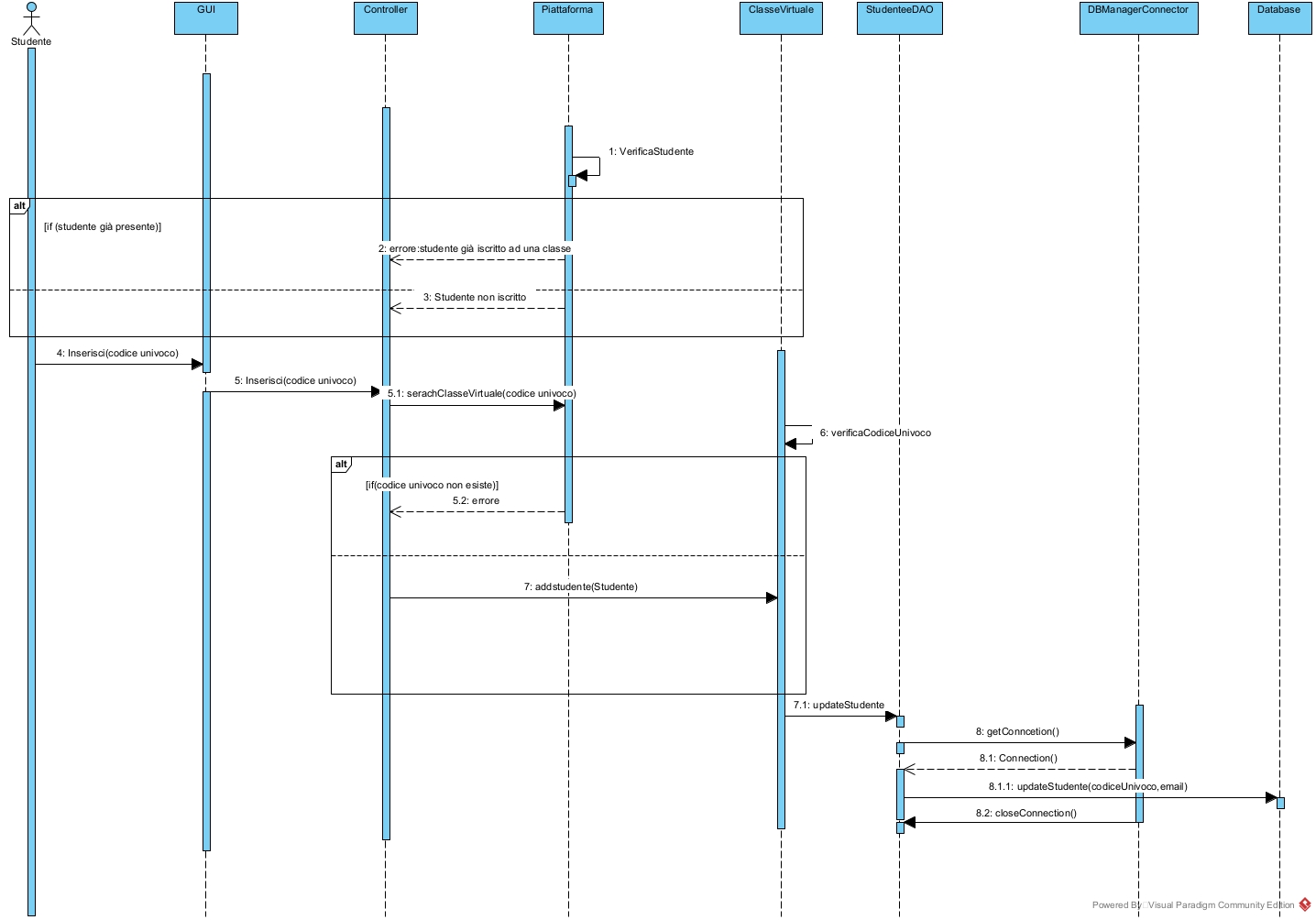
****

**4.2 Diagrammi di sequenza**

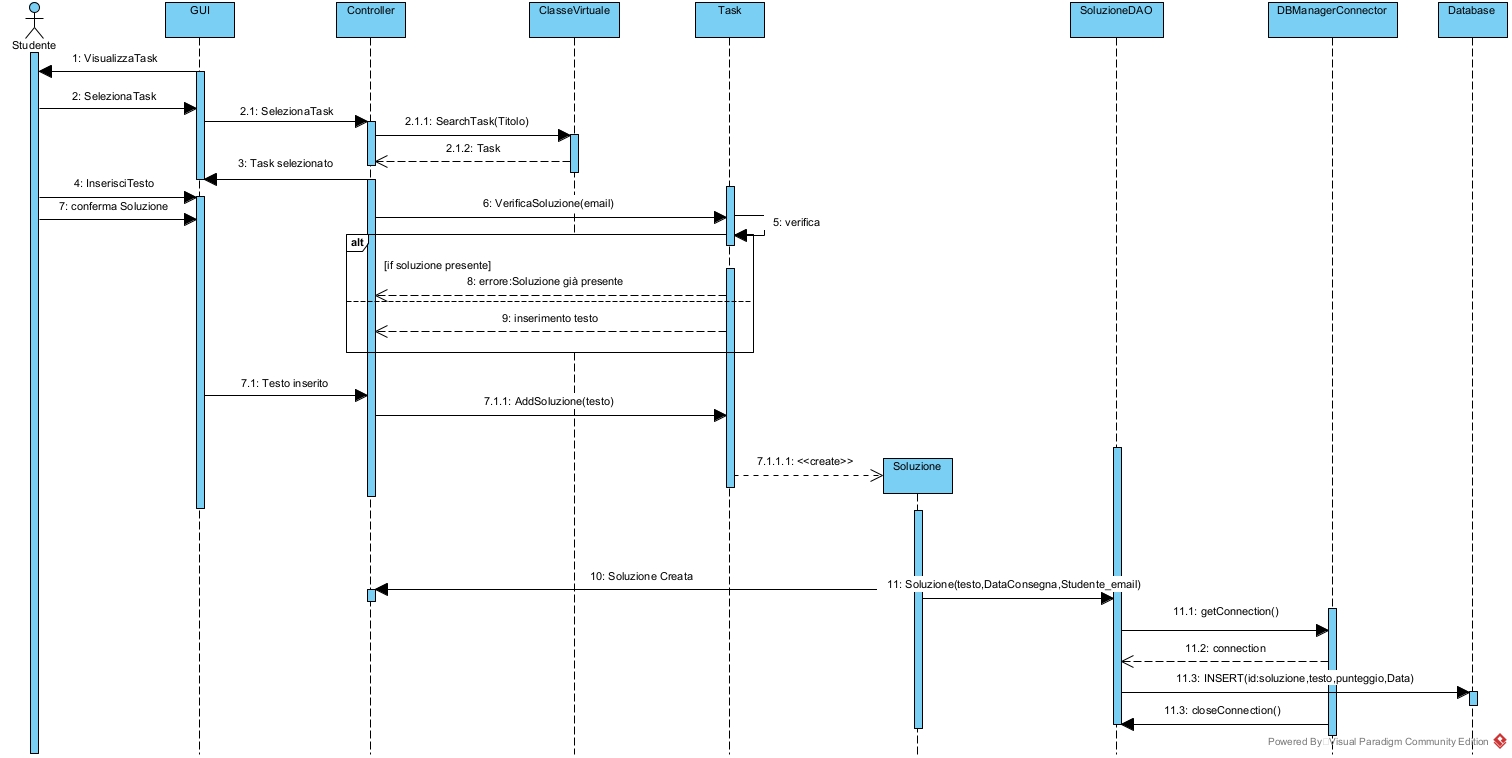
**4.2.1 Creazione Task**



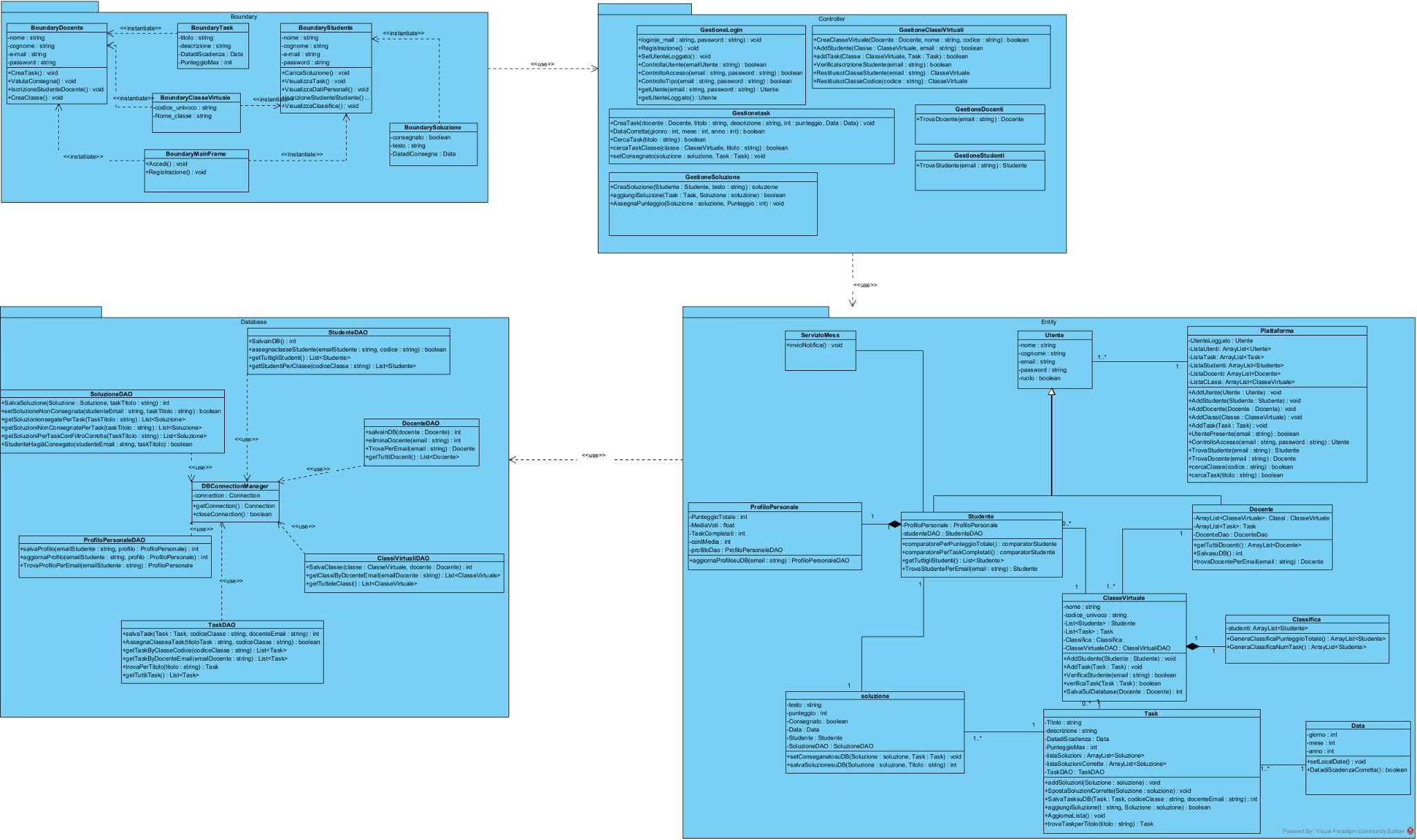
**4.2.2 Iscrizione classe Studente**



**4.2.3 Carica soluzione**



**4.3 Diagramma delle classi di Progetto**



Il Diagramma delle classi di progettazione contiene tutte le classi sviluppate con tutti gli attributi ed i metodi necessari per il software.

**5. Implementazione**

**Requisiti Del Software:**

JDK(versione 17 o superiore)

la versione 17 garantisce compatibilità con le librerie;

MySQLWorkbench 8.0

persistenza dei dati

Connettore JDBC:

file:mysql-connector-j-8.0.33

**File**

file .jar: Contengono il codice compilato,risorse e librerie esterne:

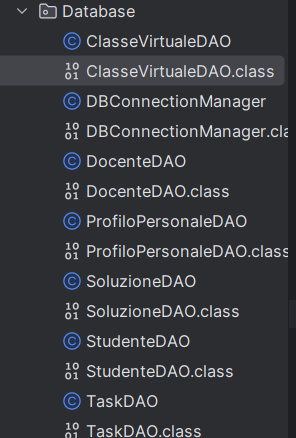
java -jar Task4Class.jar

file.class:Contengono le classi compilate

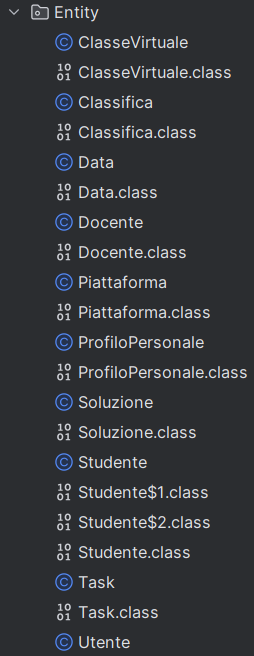
config.properties: per parametri come URL del DB,credenziali ecc..

Script SQL:Contiene la creazione delle tabelle e l’inserimento dei dati

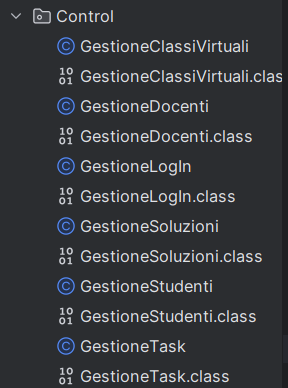
**5.1 Package Database**

****

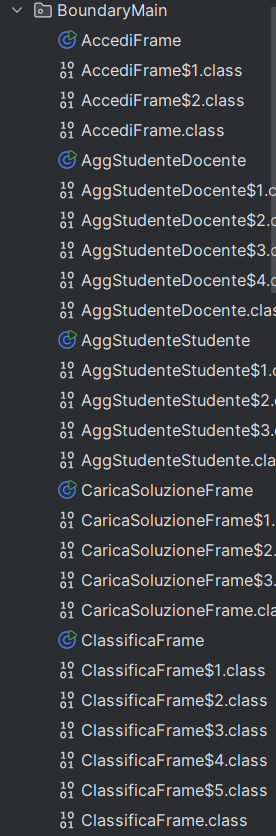
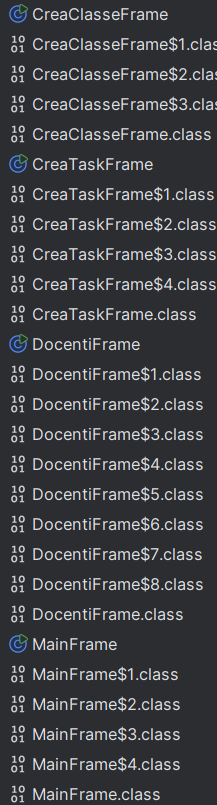
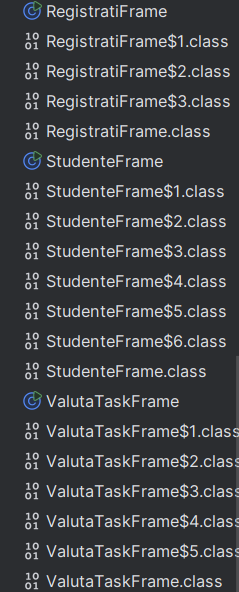
**5.2 Package Entity**

****

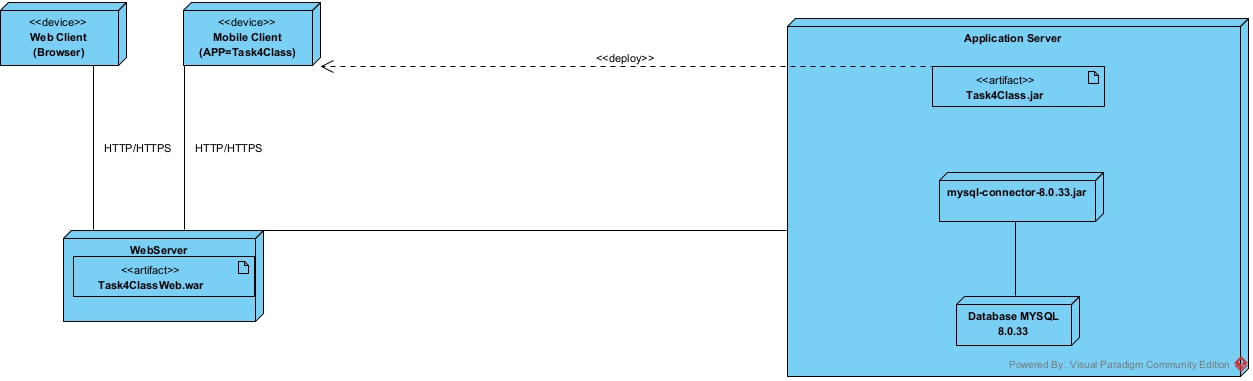
**5.3 Package Control**

****

**5.4 Package Boundary**

****

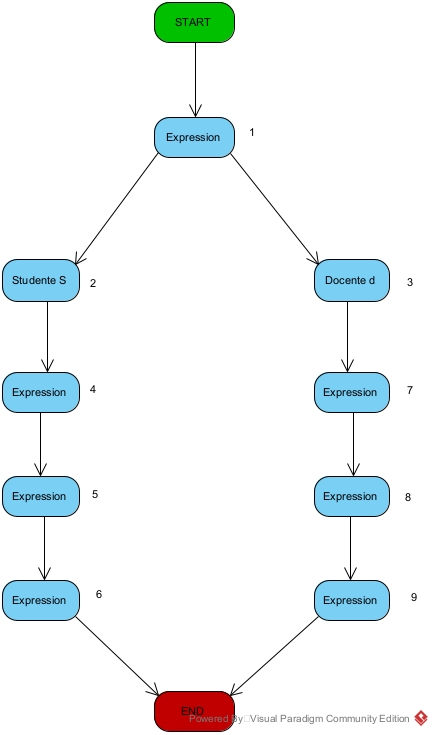
**5.5 Diagramma di Deployment**



**6. Testing**

**6.1 Test Strutturale**

**Crea Utente**

**CODICE:**

**public static String creaUtente(String nome, String cognome, String email, String password, boolean ruolo) {**

**if (ruolo) { (1)**

**Studente s = new Studente(nome, cognome, email, password, ruolo); (2)**

**Piattaforma.addUtente(s); (4)**

**Piattaforma.addStudente(s); (5)**

**return "Studente aggiunto!"; (6)**

**}else {**

**Docente d = new Docente(nome, cognome, email, password, ruolo); (3)**

**Piattaforma.addUtente(d); (7)**

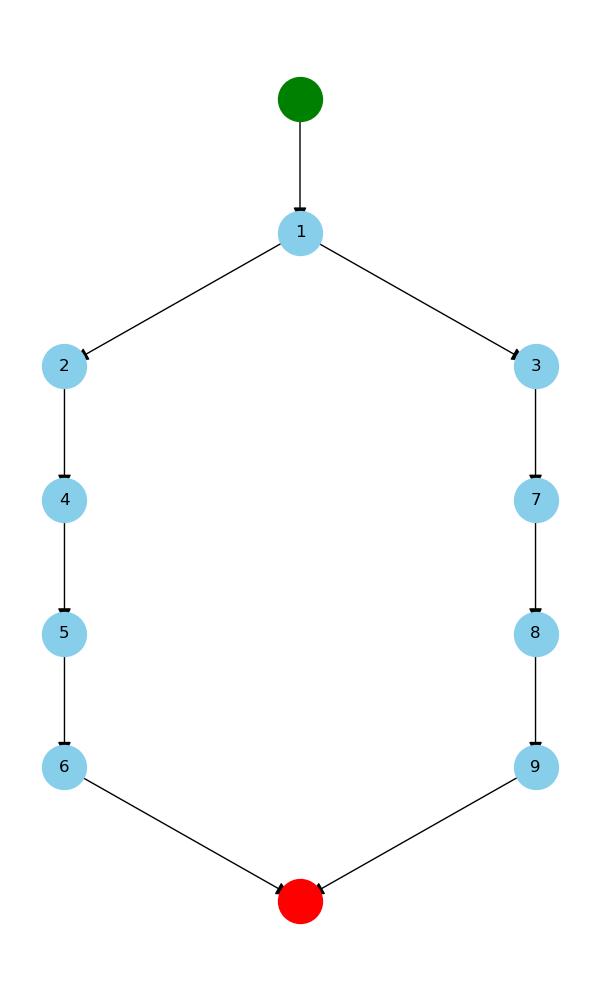
**Piattaforma.addDocente(d); (8)**

**return "Docente aggiunto!";(9)**

**}**

**}**

Accorpando i nodi in sequenza si ottiene il seguente CFG:



NUMERO CICLOMATICO:

numero di regioni chiuse del grafo = 1

numero di nodi predicati (1) +1 = 2

E-N+2=11-11+2=2

CAMMINI INDIPENDENTI:

1) 0-1-2-4-5-6 La condizione dell’if è vera

2) 0-1-3-7-8-9 La condizione dell’if è falsa

**GetTaskByClasseCodice**

Il seguente metodo è implementato nella classe TaskDAO nel package Database:

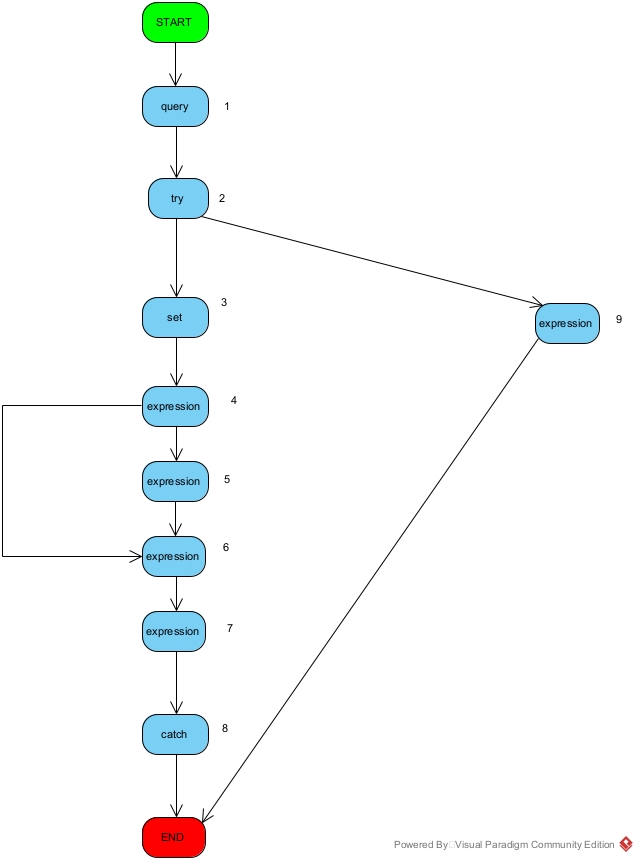
**CODICE:**

**public static List<Task> getTaskByClasseCodice(String codiceClasse) {**

**List<Task> tasks = new ArrayList<>();**

**String query = "SELECT Titolo, descrizione, Data, punteggioMax FROM Task WHERE ClasseVirtuale\_codiceunivoco = \"" + codiceClasse + "\""; (1)**

**System.out.println(query);**

**try { (2)**

**ResultSet rs = DBConnectionManager.selectQuery(query);(3)**

**while (rs.next()) { (4)**

**Date dataDB = rs.getDate("Data");**

**Data data = new Data(dataDB.getDay(), dataDB.getMonth(), dataDB.getYear()); (5)**

**Task t = new Task(**

**rs.getString("Titolo"),**

**rs.getString("descrizione"),**

**rs.getInt("punteggioMax"),**

**data(6)**

**);**

**tasks.add(t);(7)**

**}**

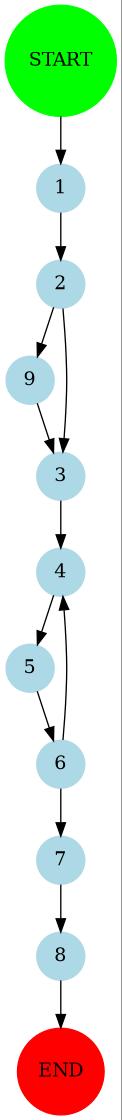
**} catch (SQLException e) (8)**

**e.printStackTrace();(9)**

**}**

**return tasks;**

**}**

Accorpando i nodi in sequenza si ottiene il seguente CFG:

NUMERO CICLOMATICO:

numero di regioni chiuse del grafo = 2

numero di nodi predicati (2) +1 = 3

M=E-N+2=10-9+2=3

CAMMINI INDIPENDENTI:

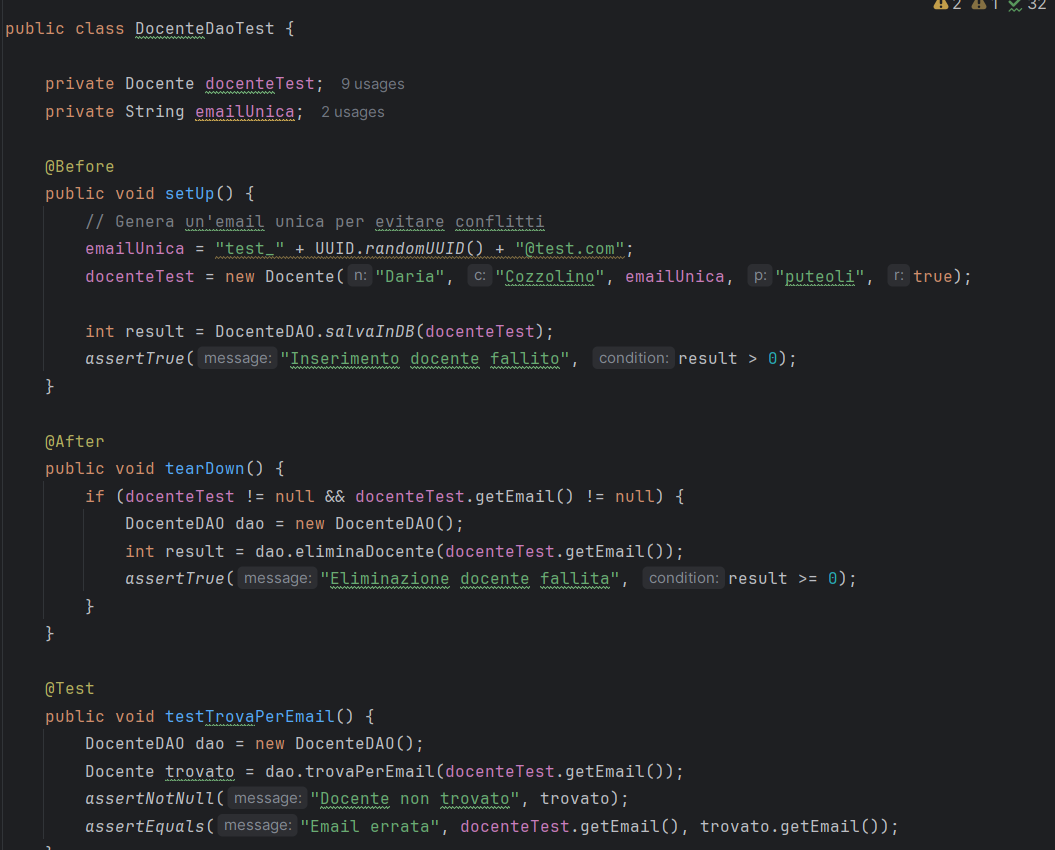
1) 0-1-2-4-5-6-4(loop)...6-7-8-9 Esecuzione del ciclo while almeno una volta;

2) 0-1-3-7-8-9 Salto del blocco try; esecuzione del ramo alternativo

3)0-1-2-9 Fallimento del blocco try; attivazione del blocco

**6.2 JUnit-Test di Unità**

**DocenteDaoTest**



**6.3 Test Funzionale**

**Creazione Task**

| **TEST SUITE** |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test case ID | Descrizione | Classi di equivalenza coperte | Pre-condizioni | Input |  | Output attesi | Post-condizioni attese | Output ottenuti | Post-condizioni ottenute | Esito (FAIL,PASS) |
| 1 | Tutti input validi | Titolo valido, Descrizione valida, Data Scadenza valida, Punteggio max valido | // | {Titolo: “Compito di italiano", Descrizione:”Tema su vita e opere di Giovanni Pascoli”, Data di scadenza:”01-02-2027”,Punteggio max: “10”} |  | Task creato | Il task compare nella lista dei task creati della classe virtuale | Task creato | Il task è stato creato e compare nella lista dei task della classe virtuale | PASS |
| 2 | Titolo già presente nel sistema | Titolo già esistente [ERROR], Descrizione valida, Data Scadenza valida, Punteggio max valido |  | {Titolo: “Compito di italiano", Descrizione:”Tema su vita e opere di Giovanni Pascoli”, Data di scadenza:”01-02-2027”,Punteggio max: “10”} |  | Titolo non valido |  | Titolo già esistente |  |  |
| 3 | Titolo stringa >50 caratteri | Titolo stringa>50 [ERROR], Descrizione,Data Scadenza, Punteggio max validi | // | {**Titolo:** “aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa”,Descrizione:”Tema su vita e opere di Giovanni Pascoli”, Data di scadenza:”01-02-2027”,Punteggio max: “10”} |  | Titolo non valido | // | Titolo non valido | Il task non è stato creato | PASS |
| 4 | Titolo stringa con caratteri speciali | Titolo stringa con simboli [ERROR],Descrizione,Data Scadenza, Punteggio max validi | // | {**Titolo**: “@#Compito”,Descrizione:”Tema su vita e opere di Giovanni Pascoli”, Data di scadenza:”01-02-2027”,Punteggio max: “10”} |  | Titolo non valido | // | Titolo non valido | Il task non è stato creato | PASS |
| 5 | Descrizione stringa vuota o maggiore di 1000 caratteri | Titolo valido, Descrizione stringa vuota o >1000 [ERROR], Data Scadenza, Punteggio max validi | // | {Titolo: “Compito di Matematica”,**Descrizione**:” ”, Data di scadenza:”01-02-2027”,Punteggio max: “10”} |  | Descrizione troppo lunga o inesistente | // |  | Il task non è stato creato | PASS |
| 6 | Data di scadenza inserita già passata | Titolo valido, Descrizione valida, Data di Scadenza passata [ERROR], Punteggio max valido | // | {Titolo: “Compito di Matematica”,Descrizione:”2+2= ”, **Data di scadenza:**”01-02-2007”,Punteggio max: “10”} |  | Data di scadenza inserita già passata | // | Data inserita passata | Il task non è stato creato | PASS |
| 7 | Formato data non valido | Titolo valido, Descrizione valida, Data non valida [ERROR], Punteggio max valido |  | {Titolo: “Compito di Matematica”,Descrizione:”2+2= ”, **Data di scadenza:**”ABCD”,Punteggio max: “10”} |  |  |  | Data inserita non valida | Il task non è stato creato | PASS |
| 8 | Punteggio max numero negativo | Titolo valido, Descrizione valida, Data scadenza valida, Punteggio max numero negativo [ERROR]. | // | {Titolo: “Compito di Matematica”,Descrizione:”2+2= ”, Data di scadenza**:**”01-02-2027”,**Punteggio max**: “-10”} |  | Punteggio max non valido | // | Punteggio non inserito | Il task non è stato creato | PASS |
| 9 | Punteggio max valore non numerico | Titolo valido, Descrizione valida, Data scadenza valida, Punteggio max valore non numerico [ERROR]. |  | {Titolo: “Compito di Matematica”,Descrizione:”2+2= ”, Data di scadenza**:**”01-02-2027”,**Punteggio max**: “ABC”} |  | Punteggio max non valido | // | Punteggio non inserito | Il task non è stato creato | PASS |

**Carica Soluzione**

| **TEST**  **SUITE** | | | | | | |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test case ID | Descrizione | Classi di equivalenza coperte | Pre-condizioni | Input | Output attesi | Post-condizioni attese | Output ottenuti | Post-condizioni ottenute | Esito (FAIL,PASS) |
| 1 | Tutti input validi | Testo valido | // | {**Testo**:“La soluzione al quesito 1 è 5.  La soluzione al quesito 2 è 67”} | Soluzione aggiunta | La soluzione viene aggiunta alla lista delle soluzioni del task | Soluzione aggiunta | La soluzione viene creata e aggiunta alla lista delle soluzioni del task | PASS |
| 2 | Stringa testo vuota o troppo lungo | Testo vuoto o di lunghezza >5000 caratteri [ERROR] | // | {**Testo**:“ ”} | Aggiungi testo o 5000 caratteri superati |  | Testo non valido | La soluzione non viene creata | PASS |
| 3 | Consegna effettuata oltre la data di scadenza | Testo valido, data oltre il termine | // | {Testo: “La soluzione al quesito 1 è 5”, **Data** **consegna**: 2025-06-13}, deadline = 2025-06-12 | Consegna “in ritardo” Soluzione aggiunta | La soluzione viene aggiunta alla lista delle soluzioni del task con flag “in ritardo” | Soluzione aggiunta | La soluzione viene creata e aggiunta alla lista delle soluzioni del task | PASS |

**Iscrizione Classe Studente**

| **TEST**  **SUITE** | | | | | | |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test case ID | Descrizione | Classi di equivalenza coperte | Pre-condizioni | Input | Output attesi | Post-condizioni attese | Output ottenuti | Post-condizioni ottenute | Esito (FAIL,PASS) |
| 1 | Tutti input validi | Codice valido | // | {**Testo**:"ABC 23"} | Aggiunto alla classe | Lo studente viene aggiunto alla classe associata al codice | Sei stato aggiunto alla classe | Lo studente viene aggiunto alla classe virtuale | PASS |
| 2 | Stringa testo vuota, troppo lungo o con caratteri speciali | Codice: Stringa vuota o > 5 o con caratteri speciali [ERROR] | // | {**Testo**:“ ”} | Aggiungi codice o codice errato |  | Codice non valido | Lo studente non viene aggiunto alla classe | PASS |
| 3 | Lo Studente è già iscritto ad una classe virtuale | Codice valido | Lo studente è iscritto ad una classe virtuale | {Testo:"ABC 23"} | Sei già iscritto ad una classe | Lo studente rimane iscritto alla classe precedente | Non puoi iscriverti a questa classe | Lo studente non viene aggiunto alla classe | PASS |