

Università degli Studi di Napoli Federico II

**Corso di Laurea in Ingegneria Informatica**

**Corso di Ingegneria del Software**

**Prof. A.R. Fasolino - A.A. 2022 - 23**

******

***Progetto***

Gestione Istituto Scolastico

“Learnopolis”

Studente: Cristina Carleo N46005492 cr.carleo@studenti.unina.it

Vincenzo Luigi Bruno N46005698 vincenzol.bruno@studenti.unina.it

Anna Flavia De Rosa N46005699 annafl.derosa@studenti.unina.it

**INDICE**

Sistema software per la gestione di un istituto scolastico

1. Specifiche informali 1

2. Analisi e specifica dei requisiti 2

2.1 Analisi nomi-verbi 2

2.2 Revisione dei requisiti 3

2.3 Glossario dei termini 3

2.4 Classificazione dei requisiti 4

2.4.1 Requisiti funzionali 4

2.4.2 Requisiti sui dati 5

2.4.3 Vincoli / Altri requisiti 5

2.5 Modellazione dei casi d’uso 6

2.5.1 Attori e casi d’uso 6

2.5.2 Diagramma dei casi d’uso 2

2.5.3 Scenari 3

2.6 Diagramma delle classi 8

2.7 Diagrammi di sequenza 11

2.8 Verifica della completezza dei requisiti 16

4. Piano di test funzionale 23

5. Progettazione 46

5.1 Diagramma delle classi 46

5.2 Diagrammi di sequenza 49

6. Implementazione 52

7. Testing 55

7.1 Test strutturale 55

7.1.1 Complessità ciclomatica 55

7.2 Test funzionale 57

# Specifiche informali

Si vuole realizzare sistema software per la gestione di un istituto scolastico.

La segreteria registra al sistema gli utenti, ovvero i docenti, gli alunni e i genitori, specificando il nome, cognome, data di nascita, codice fiscale, comune di residenza, email, numero di cellulare, username, password.

L’istituto scolastico è organizzato in classi. Ogni classe è caratterizzata dalla sezione (identificata da una lettera) e da un anno espresso in numeri romani (da 1 a 5). Per ogni classe il sistema tiene traccia del registro elettronico (uno per ciascun anno scolastico), che può essere consultato dal preside e dai docenti.

Per i docenti si vuole tenere traccia delle classi ove insegnano e delle materie insegnate in tali classi in ciascun anno scolastico. Ogni alunno è caratterizzato anche da una matricola e dalla classe frequentata.

I docenti di ogni classe devono riportare nel registro le attività (descritte come testo libero) svolte in una certa classe in una certa data. Inoltre, essi possono aggiungere i voti di ogni alunno, riportando la data. Se il voto inserito dal docente è non sufficiente (< 6) viene inviata una notifica ai genitori dell’alunno che ha ottenuto la valutazione negativa.

Gli alunni di una classe possono consultare il registro per leggere le attività svolte in una determinata data, mentre i genitori possono visionare la media voti del proprio figlio nell’ultimo quadrimestre più il dettaglio dei singoli voti in tutte le materie.

Al termine di ogni quadrimestre, il sistema genera la pagella di ogni alunno riportando la media voti per ogni materia. I docenti possono apportare delle modifiche alle valutazioni relative ai propri insegnamenti arrotondando il voto per eccesso o per difetto.

Appena creata, la pagella si trova nello stato “non approvata”: è compito del preside approvare la pagella. La pagella non può essere modificata dai docenti una volta approvata.

All’atto dell’approvazione, se tutte le pagelle associate agli alunni della classe sono state approvate, il sistema notifica tutti i genitori degli alunni, inviando loro una email.

# Analisi e specifica dei requisiti

## Analisi nomi-verbi

Si vuole realizzare sistema software per la gestione di un istituto scolastico.

La segreteria registra al sistema gli utenti, ovvero i docenti, gli alunni e i genitori, specificando il nome, cognome, data di nascita, codice fiscale, comune di residenza, email, numero di cellulare, username, password.

L’istituto scolastico è organizzato in classi. Ogni classe è caratterizzata dalla sezione (identificata da una lettera) e da un anno espresso in numeri romani (da 1 a 5). Per ogni classe il sistema tiene traccia del registro elettronico (uno per ciascun anno scolastico), che può essere consultato dal preside e dai docenti.

Per i docenti si vuole tenere traccia delle classi ove insegnano e delle materie insegnate in tali classi in ciascun anno scolastico. Ogni alunno è caratterizzato anche da una matricola e dalla classe frequentata.

I docenti di ogni classe devono riportare nel registro le attività (descritte come testo libero) svolte in una certa classe in una certa data. Inoltre, essi possono aggiungere i voti di ogni alunno, riportando la data. Se il voto inserito dal docente è non sufficiente (< 6) viene inviata una notifica ai genitori dell’alunno che ha ottenuto la valutazione negativa.

Gli alunni di una classe possono consultare il registro per leggere le attività svolte in una determinata data, mentre i genitori possono visionare la media voti del proprio figlio nell’ultimo quadrimestre più il dettaglio dei singoli voti in tutte le materie.

Al termine di ogni quadrimestre, il sistema genera la pagella di ogni alunno riportando la media voti per ogni materia. I docenti possono apportare delle modifiche alle valutazioni relative ai propri insegnamenti arrotondando il voto per eccesso o per difetto.

Appena creata, la pagella si trova nello stato “non approvata”: è compito del preside approvare la pagella. La pagella non può essere modificata dai docenti una volta approvata.

All’atto dell’approvazione, se tutte le pagelle associate agli alunni della classe sono state approvate, il sistema notifica tutti i genitori degli alunni, inviando loro una email.

*LEGENDA:  
Classe  
Attributo  
Funzionalità  
Attore*

*Classe-Attore*

## Revisione dei requisiti

1. *La segreteria registra al sistema gli utenti, ovvero i docenti, gli alunni e i genitori.*
2. *Degli utenti viene specificato il nome, cognome, data di nascita, codice fiscale, comune di residenza, email, numero di cellulare, username, password.*
3. *L’istituto scolastico è organizzato in classi.*
4. *Ogni classe è caratterizzata dalla sezione e da un anno espresso in numeri romani*
5. *Per ogni classe il sistema tiene traccia del registro elettronico, che può essere consultato dal preside e dai docenti.*
6. *Per i docenti si vuole tenere traccia delle classi ove insegnano e delle materie insegnate in tali classi in ciascun anno scolastico.*
7. *Ogni alunno è caratterizzato anche da una matricola e dalla classe frequentata.*
8. *I docenti di ogni classe devono riportare nel registro le attività svolte in una certa classe in una certa data.*
9. *I docenti possono aggiungere i voti di ogni alunno, riportando la data.*
10. *Se il voto inserito dal docente è non sufficiente (< 6) viene inviata una notifica ai genitori dell’alunno che ha ottenuto la valutazione negativa.*
11. *Gli alunni di una classe possono consultare il registro per leggere le attività svolte in una determinata data.*
12. *I genitori possono visionare la media voti del proprio figlio nell’ultimo quadrimestre più il dettaglio dei singoli voti in tutte le materie.*
13. *Al termine di ogni quadrimestre, il sistema genera la pagella di ogni alunno riportando la media voti per ogni materia.*
14. *I docenti possono apportare delle modifiche alle valutazioni relative ai propri insegnamenti arrotondando il voto per eccesso o per difetto.*
15. *Appena creata, la pagella si trova nello stato “non approvata”*
16. *É compito del preside approvare la pagella.*
17. *La pagella non può essere modificata dai docenti una volta approvata.*
18. *All’atto dell’approvazione, se tutte le pagelle associate agli alunni della classe sono state approvate, il sistema notifica tutti i genitori degli alunni, inviando loro una email.*

## Glossario dei termini

|  |  |  |
| --- | --- | --- |
| Termine | Descrizione | Sinonimi |
| Classe | Gruppo di alunni caratterizzati dalla sezione e a cui è associato un registro elettronico per ciascun anno scolastico e un insieme di docenti per ogni anno. |  |
| Registro elettronico | Oggetto che tiene traccia delle attività svolte dalla classe |  |
| Attività | Breve descrizione di ciò che svolge una classe in una certa data |  |
| Pagella | Documento che riporta per ogni alunno la media dei suoi voti in ogni materia, che può essere modificato dai docenti solo prima dell’approvazione del preside |  |
| Quadrimestre | Periodo di quattro mesi al termine del quale viene generata una pagella. (Settembre - Gennaio, Febbraio-Giugno) |  |
| Voto | Punteggio da 1-10 assegnato all’alunno da parte di un docente | Valutazione |
| Voto non sufficiente | Voto inferiore al 6 |  |
| Genitore | Assumeremo di avere un solo account genitore per alunno, al genitore arriveranno tutte le notifiche relative all’alunno corrispondente |  |

## Classificazione dei requisiti

## Requisiti funzionali

|  |  |  |
| --- | --- | --- |
| ID | Requisito | Origine (n. frase dei requisiti revisionati) |
| RF01 | Il sistema deve offrire alla Segreteria una funzionalità per registrare docenti, alunni e genitori. | 1 |
| RF02 | Il sistema deve offrire al Preside una funzionalità per consultare il registro elettronico | 5 |
| RF03 | Il sistema deve offrire ai docenti una funzionalità per consultare il registro elettronico | 5 |
| RF04 | Il sistema deve offrire ai docenti una funzionalità per riportare nel registro le attività svolte in una certa classe in una certa data. | 8 |
| RF05 | Il sistema deve offrire ai docenti una funzionalità per aggiungere i voti di ogni alunno, riportando la data. | 9 |
| RF06 | Il sistema deve inviare una notifica ai genitori dell’alunno nel caso in cui questo abbia ottenuto una valutazione negativa. | 10 |
| RF07 | Il sistema deve offrire all’alunno una funzionalità per consultare il registro per leggere le attività svolte in una determinata data. | 11 |
| RF08 | Il sistema deve effettuare automaticamente la media dei voti per ogni alunno nell’ultimo quadrimestre. | 12 |
| RF09 | Il sistema deve offrire al genitore una funzionalità per visionare la media voti del proprio figlio nell’ultimo quadrimestre più il dettaglio dei singoli voti in tutte le materie. | 12 |
| RF10 | Il sistema deve generare automaticamente al termine di ogni quadrimestre la pagella di ogni alunno riportando la media dei voti per ogni materia. | 13 |
| RF11 | Il sistema deve offrire ai docenti una funzionalità per apportare delle modifiche alle valutazioni relative ai propri insegnamenti arrotondando il voto per eccesso o per difetto. | 14 |
| RF12 | Il sistema deve inserire automaticamente la pagella generata nello stato “non approvata”. | 15 |
| RF13 | Il sistema deve offrire al preside una funzionalità per approvare la pagella. | 16 |
| RF14 | Il sistema deve impedire che la pagella possa essere modificata dai docenti una volta approvata. | 17 |
| RF15 | Il sistema deve inviare una mail a tutti i genitori degli alunni all’atto dell’approvazione, nel caso in cui tutte le pagelle associate agli alunni della classe siano state approvate | 18 |

## Requisiti sui dati

|  |  |  |
| --- | --- | --- |
| ID | Requisito | Origine (n. frase dei requisiti revisionati) |
| RD01 | Degli utenti viene specificato il nome, cognome, data di nascita, codice fiscale, comune di residenza, email, numero di cellulare, username, password. | 2 |
| RD02 | L’istituto scolastico è organizzato in classi. | 3 |
| RD03 | Ogni classe è caratterizzata dalla sezione caratterizzata da una lettera e da un anno espresso in numeri romani | 4 |
| RD04 | Per ogni classe il sistema tiene traccia del registro elettronico | 5 |
| RD05 | Il sistema tiene traccia di un registro elettronico per classe per ciascun anno scolastico |  |
| RD06 | Per i docenti si vuole tenere traccia delle classi ove insegnano e delle materie insegnate in ciascun anno scolastico | 6 |
| RD07 | Ogni alunno è caratterizzato anche da una matricola e dalla classe frequentata. | 7 |
| RD08 | Ogni attività, descritta come testo libero deve essere corredata della classe e della data nella quale è stata svolta | 8 |
| RD09 | Ogni voto inserito deve riportare una data | 9 |
| RD10 | Ad ogni pagella è associato uno stato che può assumere il valore di “approvata” se il preside l’ha approvata, “non approvata” in caso contrario | 15 |

## Vincoli / Altri requisiti

|  |  |  |
| --- | --- | --- |
| ID | Requisito | Origine (n. frase dei requisiti revisionati) |
| V01 | Ogni quadrimestre dura quattro mesi |  |
| RNF01 | Per l’invio della mail di notifica, deve essere disponibile un server di posta elettronica esterno al sistema |  |

## Modellazione dei casi d’uso

## Attori e casi d’uso

***Attori Primari:***

* Docenti
* Alunni
* Genitori
* Segreteria
* Preside

***Attori Secondari:***

* Servizio email

***Casi d’uso:***

* UC1: RegistraUtente
* UC2: InserisciClasse
* UC3: VisualizzaRegistro
* UC4: RiportaAttività
* UC5: AggiungiVoto
* UC6: ArrotondaVoto
* UC7: ApprovaPagella

***Casi d’ uso di inclusione:***

* UC8: AggiornaPagella
* UC9: VisualizzaAttività
* UC10: VisualizzaVoti

***Casi d’ uso di estensione:***

* UC11: InviaNotifica
* UC12: NotificaInsufficenza
* UC13: NotificaPagella

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Caso d’uso | | | | Attori Primari | | Attori Secondari | Incl. / Ext | Requisiti corrispondenti |
| UC1: RegistraUtente | | Segreteria | | | | - | - | RF01 |
| UC2: InserisciClasse | | | Segreteria | | | - | - | - |
| UC3: VisualizzaRegistro | | | | Docente, Preside | | - | - | RF02, RF03 |
| UC4: RiportaAttività | Docente | | | | | - | - | RF04 |
| UC5: AggiungiVoto | | | | Docente | | - | Extend InviaNotifica | RF05 |
| UC8: ModificaPagella | | | | - | | - | Extend InviaNotifica | RF10, RF11, RF12, RF13, RF14 |
| UC9: VisualizzaAttività | | | | Alunno | | - | - | RF07 |
| UC10: VisualizzaVoti | | | | Genitore | | - | - | RF8, RF09 |
| UC11: InviaNotifica | | | | Segreteria | | - | Estensione di AggiungiVoto, AggiornaPagella | RF06, RF15 |
| UC12: NotificaInsufficienza | | | | | - | Servizio email | Estensione di InviaNotifica | RF06 |
| UC13: NotificaPagella | | | | - | | Servizio email | Estensione di InviaNotifica | RF15 |

## Diagramma dei casi d’uso

Dall’analisi dei casi d’uso è stato prodotto il seguente diagramma UML, che rappresenta i casi d’uso individuati, le interazioni tra gli attori e il sistema GestioneIstitutoScolastico, e le relazioni tra casi d’uso, in particolare quelle di uso (o inclusione), che formalizzano il caso in cui più casi d’uso comprendono una serie di azioni comuni, e di estensione, che formalizzano sequenze opzionali di eventi o casi eccezionali che partono da un caso d’uso base.

Immagine che contiene testo, diagramma, schermata, linea

Descrizione generata automaticamente

## Scenari

Di seguito è descritto il *main success scenario* con relative estensioni ed inclusioni

|  |  |  |  |
| --- | --- | --- | --- |
| Caso d’uso: | | RegistraUtente | |
| Attore primario | | Segreteria | |
| Attore secondario | | - | |
| Descrizione | | La segreteria ha il compito di registrare nuovi utenti alla piattaforma | |
| Pre-Condizioni | | L’istituto è stato definito | |
| Sequenza di eventi  principale | 1. Il caso d’uso inizia con la richiesta della segreteria di registrare un utente 2. La Segreteria specifica il ruolo, nome, cognome, data di nascita, codice fiscale, comune di residenza, email, numero di cellulare, username e password degli utenti. 3. L’utente viene creato 4. Il caso d’uso termina | | |
| Post-Condizioni | | L’utente viene aggiunto al sistema coi relativi privilegi | |
| Casi d’uso correlati | | *-* | |
| Sequenza di eventi alternativi | | | Al punto 3:   * 1. Se si tratta di un Docente verranno specificate le materie insegnate nelle varie classi      1. Il sistema controlla che le materie siano valide, in caso contrario invia un messaggio di ERRORE   2. Se si tratta di un Genitore, verrà specificata la matricola dello studente di cui è genitore.      1. Il sistema controlla che la matricola inserita sia valida, altrimenti restituisce un messaggio di errore.   3. Se si tratta di un Alunno, il sistema gli assegna una matricola e verrà specificata la classe di appartenenza.      1. Il sistema controlla che la classe sia valida, in caso contrario invia un messaggio di ERRORE. |

|  |  |  |  |
| --- | --- | --- | --- |
| Caso d’uso: | | InserisciClasse | |
| Attore primario | | Segreteria | |
| Attore secondario | | - | |
| Descrizione | | La segreteria ha il compito di inserire le classi di cui si compone l’istituto | |
| Pre-Condizioni | | - | |
| Sequenza di eventi  principale | 1. La Segreteria specifica anno e sezione della classe 2. Il Sistema istanzia la classe.    1. Se l’operazione va a buon fine, manda una un messaggio di CLASSE CREATA.    2. Se l’operazione non va a buon fine, manda un messaggio di ERRORE 3. La Segreteria inserisce il nome della materia per ogni insegnamento della classe 4. Il sistema istanzia la classe    1. Se l’operazione va a buon fine, il sistema manda una un messaggio di MATERIA AGGIUNTA.    2. Se l’operazione non va a buon fine, il sistema manda un messaggio di ERRORE 5. Il caso d’uso termina. | | |
| Post-Condizioni | | Classe creata | |
| Casi d’uso correlati | | *-* | |
| Sequenza di eventi alternativi | | | - |

|  |  |  |
| --- | --- | --- |
| Caso d’uso: | | VisualizzaRegistro |
| Attore primario | | Docente, Preside |
| Attore secondario | | - |
| Descrizione | | Il sistema offre al preside e ai docenti una funzionalità per visualizzare il registro di una certa classe |
| Pre-Condizioni | | L’istituto è stato definito, l’utente ha effettuato il login |
| Sequenza di eventi  principale | 1. L’utente inserisce la classe di cui vuole visualizzare il registro    1. Il sistema verifica che la classe esista, in caso contrario restituisce un messaggio di ERRORE 2. L’utente inserisce una data 3. Il sistema manda a video la descrizione delle attività relative a quella data    1. Se non ci sono attività per quella data il sistema manda un messaggio di ERRORE 4. Il caso d’uso termina | |
| Post-Condizioni | | - |
| Casi d’uso correlati | | *-* |
| Sequenza di eventi  alternativi | | - |

|  |  |  |  |
| --- | --- | --- | --- |
| Caso d’uso: | | RiportaAttività | |
| Attore primario | | Docente | |
| Attore secondario | | - | |
| Descrizione | | Il sistema offre al docente la possibilità di riportare un’attività sul registro | |
| Pre-Condizioni | | L’istituto è stato definito, il docente ha effettuato il login | |
| Sequenza di eventi  principale | 1. Il Docente seleziona la classe a cui vuole aggiungere l’attività    1. Il sistema verifica che la classe esista, in caso contrario restituisce un messaggio di ERRORE 2. Il Docente inserisce la data in cui vuole aggiungere un’attività    1. Il sistema verifica che la data esista e sia inclusa nel quadrimestre corrente, in caso contrario restituisce un messaggio di ERRORE 3. Il Docente inserisce la descrizione relativa all’attività 4. Il sistema istanzia l’attività    1. Se l’operazione va a buon fine, il sistema manda una un messaggio di ATTIVITÁ CREATA.    2. Se l’operazione non va a buon fine, il sistema manda un messaggio di ERRORE 5. Il caso d’uso termina | | |
| Post-Condizioni | | Attività creata | |
| Casi d’uso correlati | | *-* | |
| Sequenza di eventi alternativi | | | - |

|  |  |  |  |
| --- | --- | --- | --- |
| Caso d’uso: | | AggiungiVoto | |
| Attore primario | | Docente | |
| Attore secondario | | - | |
| Descrizione | | Il docente inserisce un voto ad un alunno in una certa data e in una certa materia | |
| Pre-Condizioni | | L’istituto è stato definito, l’utente ha effettuato login. | |
| Sequenza di eventi  principale | 1. Il Docente inserisce la materia nella quale vuole aggiungere una valutazione    1. Il sistema verifica che la materia esista e sia insegnata da quel docente, in caso contrario restituisce un messaggio di ERRORE 2. Il Docente inserisce la data dell’interrogazione    1. Il sistema verifica che la data sia compresa nel quadrimestre e non sia successiva alla data corrente, in caso contrario restituisce un messaggio di ERRORE 3. Il Docente inserisce l’alunno a cui vuole aggiungere il voto    1. Il sistema verifica che la l’alunno esista e che appartenga alla classe in cui il docente insegna la materia selezionata, in caso contrario restituisce un messaggio di ERRORE 4. Il Docente inserisce la valutazione    1. Il sistema verifica che la valutazione sia compresa tra 1 e 10, in caso contrario restituisce un messaggio di ERRORE    2. Se la valutazione è una valutazione negativa il sistema invia una email al genitore dell’alunno, questa funzione è gestita da *InviaNotifica* 5. L’utente esce e il caso d’uso termina | | |
| Post-Condizioni | | Voto inserito, Email inviata | |
| Casi d’uso correlati | | *InviaNotifica. NotificaInsufficienza* | |
| Sequenza di eventi alternativi | | | - |

|  |  |  |
| --- | --- | --- |
| Caso d’uso: | | VisualizzaVoti |
| Attore primario | | Genitore |
| Attore secondario | | - |
| Descrizione | | Il sistema offre al genitore una funzionalità per visionare la media voti del proprio figlio nell’ultimo quadrimestre più il dettaglio dei singoli voti in tutte le materie. |
| Pre-Condizioni | | L’utente è un genitore che ha effettuato il login |
| Sequenza di eventi  principale | 1. Il sistema estrapola la matricola dell’alunno 2. Il sistema accede ai voti dell’alunno 3. Il sistema mostra le singole valutazioni dell’alunno e la media complessiva 4. Il caso d’uso termina | |
| Post-Condizioni | | - |
| Casi d’uso correlati | | *-* |
| Sequenza di eventi  alternativi | | - |

|  |  |  |
| --- | --- | --- |
| Caso d’uso: | | ModificaPagella |
| Attore primario | | Tempo |
| Attore secondario | | Docente, Preside |
| Descrizione | | La pagella viene aggiornata ogni volta che un docente arrotonda il voto e se il preside la approva. Quando tutte le pagelle di una classe vengono approvate il sistema invia una notifica ai genitori. |
| Pre-Condizioni | | Il sistema riceve una richiesta di modifica o approvazione della pagella, l’utente ha effettuato il login |
| Sequenza di eventi  principale | 1. L’Utente inserisce la classe a cui appartiene l’alunno per aprire il registro di riferimento    1. Il sistema verifica che la classe esista e vi insegni, in caso contrario restituisce un messaggio di ERRORE 2. L’Utente inserisce l’alunno di cui vuole modificare la pagella    1. Il sistema verifica che la l’alunno esista, in caso contrario restituisce un messaggio di ERRORE    2. Il sistema verifica che l’alunno sia della classe selezionata, in caso contrario restituisce un messaggio di ERRORE 3. Il sistema verifica lo stato della pagella relativa a quello studente    1. Se l’alunno non ha una pagella associata, questa viene creata e lo stato viene posto a “non approvata”.    2. Se la pagella si trova nello stato di “non approvata” il caso d’uso continua    3. Se la pagella si trova nello stato di “approvata”, il sistema restituisce un messaggio di ERRORE. 4. Se il sistema ha ricevuto una richiesta di modifica da parte di un Docente, il sistema aggiorna la media dei voti dell’alunno in quella materia e manda un messaggio di PAGELLA AGGIORNATA. 5. Se il sistema ha ricevuto una richiesta di approvazione, il sistema modifica lo stato della pagella e manda un messaggio di PAGELLA APPROVATA. 6. Se il numero di pagelle approvate nella classe coincide con il numero di alunni, il sistema richiama *InviaNotifica* 7. Il caso d’uso termina | |
| Post-Condizioni | | Pagella aggiornata, Pagella approvata |
| Casi d’uso correlati | | *InviaNotifica* |
| Sequenza di eventi  alternativi | | - |

|  |  |  |
| --- | --- | --- |
| Caso d’uso: | | VisualizzaAttività |
| Attore primario | | Alunno |
| Attore secondario | | - |
| Descrizione | | Il sistema offre all’alunno una funzionalità per visualizzare le attività nel registro della sua classe |
| Pre-Condizioni | | L’utente è uno studente che ha effettuato il login |
| Sequenza di eventi  principale | 1. Il sistema estrapola la classe dell’alunno 2. Il sistema accede al registro della classe dell’alunno 3. Il sistema manda a video l’elenco di attività inserite 4. L’utente inserisce una data 5. Il sistema manda a video la descrizione delle attività relative a quella data    1. Se non ci sono attività per quella data il sistema manda un messaggio di ERRORE 6. L’utente esce e il caso d’uso termina | |
| Post-Condizioni | | - |
| Casi d’uso correlati | | *-* |
| Sequenza di eventi  alternativi | | 3. Se non ci sono attività il sistema manda un messaggio di ERRORE |

|  |  |  |  |
| --- | --- | --- | --- |
| Caso d’uso: | | InviaNotifica | |
| Attore primario | | Servizio email | |
| Attore secondario | | - | |
| Descrizione | | Il sistema manda una notifica ai genitori nel caso in cui il loro figlio abbia avuto un’insufficienza o se tutte le pagelle della classe sono state approvate | |
| Pre-Condizioni | | Il sistema ha ricevuto una richiesta di inviare una notifica | |
| Sequenza di eventi  principale | 1. Se la richiesta viene da *AggiungiVoto*     1. Sarà vera la condizione VotoInsufficiente e il sistema inoltra la richiesta a *NotificaInsufficienza* 2. Se la richiesta viene da *AggiornaPagella*    1. Sarà vera la condizione di *PagelleApprovate* e il sistema inoltra la richiesta a *NotificaPagella* 3. Il caso d’uso termina | | |
| Post-Condizioni | | Notifica Inviata | |
| Casi d’uso correlati | | *AggiungiVoto, AggiornaPagella, NotificaInsufficienza, NotificaPagella* | |
| Sequenza di eventi alternativi | | | - |

|  |  |  |  |
| --- | --- | --- | --- |
| Caso d’uso: | | NotificaInsufficenza | |
| Attore primario | | Servizio email | |
| Attore secondario | | - | |
| Descrizione | | Il sistema invia una notifica ai genitori dell’alunno per notificare l’insufficienza | |
| Pre-Condizioni | | *VotoInsufficiente* | |
| Sequenza di eventi  principale | 1. Il sistema cerca il genitore corrispondente alla matricola dell’alunno che ha ricevuto un’insufficienza e prende l’email 2. Il sistema scrive una email il cui oggetto sarà “Notifica Insufficienza” e nel corpo specificherà voto, materia e data e la invia    1. Se l’operazione va a buon fine, il sistema manda un messaggio di MESSAGGIO INVIATO. 3. Il caso d’uso termina | | |
| Post-Condizioni | | Il messaggio è stato inviato | |
| Casi d’uso correlati | | *InviaNotifica* | |
| Sequenza di eventi alternativi | | | 4.1. Se l’operazione non va a buon fine, il sistema manda un messaggio di ERRORE |

|  |  |  |
| --- | --- | --- |
| Caso d’uso: | | NotificaPagella |
| Attore primario | | Servizio email |
| Attore secondario | | - |
| Descrizione | | Il sistema invia una notifica ai genitori degli alunni per notificare l’arrivo delle pagelle |
| Pre-Condizioni | | *PagelleApprovate* |
| Sequenza di eventi  principale | 1. Il sistema prende dalla classe le matricole di tutti gli alunni 2. Il sistema cerca i genitori collegati a quelle matricole e prende le email 3. Il sistema scrive una email il cui oggetto sarà “Notifica Pagella” e nel corpo specificherà la classe, il quadrimestre e l’anno scolastico. 4. Il sistema invia il messaggio di posta elettronica grazie alla casella di posta elettronica    1. Se l’operazione va a buon fine, il sistema manda un messaggio di MESSAGGI INVIATI. 5. Il caso d’uso termina | |
| Post-Condizioni | | I messaggi sono stati inviati |
| Casi d’uso correlati | | *InviaNotifica* |
| Sequenza di eventi  alternativi | | 4.1. Se l’operazione non va a buon fine, il sistema manda un messaggio di ERRORE |

## Diagramma delle classi

Per il diagramma delle classi di analisi sono state fatte le seguenti assunzioni:

* A ciascuno studente è associato un unico account Genitore, a cui possono accedere entrambi i genitori (associazione uno a uno).
* Una classe contiene un numero indefinito di studenti con relazione di aggregazione (*contenimento lasco*).
* Il registro elettronico, invece, contiene un numero indefinito di attività con relazione di composizione (*contenimento stretto*).
* L’associazione uno a molti tra MateriaInsegnata a Docente è dovuta al fatto che con la classe MateriaInsegnata intendiamo la materia insegnata da uno specifico docente in una specifica classe.

Immagine che contiene testo, diagramma, schermata, Parallelo

Descrizione generata automaticamente

A partire da questo Class Diagram semplificato è stato fatto un raffinamento.

Infatti, sono state specificate le responsabilità delle principali operazioni dedotte dall'analisi dei casi d'uso per le classi già inserite nel precedente Class Diagram, ed è stata

aggiunta un'ulteriore classe Istituto, alla quale sono state attribuite le responsabilità di registrare nuovi utenti e di istanziare le classi.

Immagine che contiene testo, diagramma, schermata, Parallelo

Descrizione generata automaticamente

L’Istituto scolastico tiene traccia di tutti gli utenti della piattaforma e si compone di varie classi. A questa classe contenitore è stata data quindi la responsabilità di registrare gli utenti della piattaforma, inserire nuove classi ed effettuare la ricerca sulle classi.

Per ogni utente il sistema tiene traccia di nome, cognome, data di nascita, codice fiscale, comune di residenza, email, numero di cellulare, username, password.

Per ogni genitore viene specificato lo studente di cui è il genitore, fra le due classi ci sarà quindi un’*associazione Uno-a-Uno*.

Ogni studente potrà essere iscritto a una o più classi nel corso della sua carriera scolastica, in quanto la singola classe a cui risulta iscritto sarà relativa a un singolo anno scolastico.

Fra le due classi sussiste quindi una relazione di aggregazione (accoppiamento lasco).

Ad ogni studente sono associate più pagelle, una per ogni quadrimestre, e ognuna si compone di più giudizi, uno per ogni materia. GiudizioFinale sarà quindi una classe associativa che si interpone fra Pagella e MateriaInsegnata. Infine, ogni studente tiene traccia dei suoi voti.

Ogni classe contiene più studenti ed è relativa a più materie, ma per ogni classe c’è un solo registro elettronico che è una composizione di attività (*accoppiamento stretto*).

Ogni attività è relativa ad un solo registro e ad un solo docente. Ogni docente può scrivere invece più attività sul registro e può inoltre insegnare una o più materie.

Ogni materia insegnata considerata nel Class Diagram è relativa ad un solo docente e ad una sola classe, questa scelta è stata realizzata cosicché da questa si potesse risalire univocamente all’una e all’altra (es. Database insegnata dal prof. Moscato nella VB del 2023). Ad ogni materia sono associati più voti, quindi la classe Valutazione sarà invece relativa ad un’unica materia e ad un unico studente.

La relazione di generalizzazione-specializzazione rappresentata nel diagramma verrà poi trasformata eliminando la superclasse e portando i suoi attributi nelle classi figlie, in quanto le sottoclassi hanno per loro natura una specificità che è opportuno preservare.

## Diagrammi di sequenza

Di seguito sono riportati i diagrammi di sequenza per alcuni dei casi d’uso che sono stati precedentemente individuati.

Per il caso d’uso *RegistraUtente*, sono stati realizzati tre diagrammi di sequenza: uno per ciascuna tipologia di utente.

* *RegistraDocente*: Prima di registrare un docente viene fatto un controllo sulle materie inserite: solo se queste esistono sarà possibile inserire i nuovi docenti nel Database in modo consistente.

Immagine che contiene testo, diagramma, Parallelo, linea

Descrizione generata automaticamente

* *RegistraStudente*: Prima di registrare uno studente viene fatto un controllo sulla classe scelta: solo se questa esiste sarà possibile inserire il nuovo studente nel Database in modo consistente.

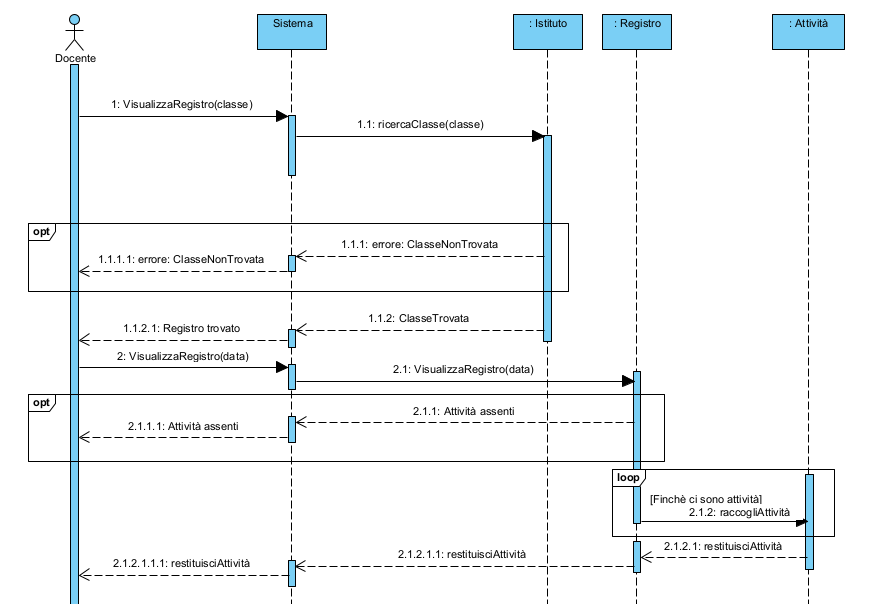
Immagine che contiene testo, diagramma, schermata, Parallelo

Descrizione generata automaticamente

* *RegistraGenitore*: prima di registrare un genitore viene fatto controllo sulla matricola dello studente figlio: solo se questa esiste e non è ancora associata ad un genitore sarà possibile inserire il nuovo genitore nel Database in modo consistente.  
  Immagine che contiene testo, schermata, diagramma, linea

  Descrizione generata automaticamente
* *VisualizzaRegistro*

Affinché un docente possa visualizzare un registro, è necessario fare un controllo sull’esistenza della classe: solo nel caso in cui la classe esista, se sono presenti attività per una certa data, queste saranno mostrate al docente.



* *AggiungiVoto*

Affinché un docente possa aggiungere una valutazione relativa ad una determinata materia in una certa data per uno studente, sono necessari controlli sulla validità di materia, data e studente: solo nel caso in cui tali controlli vadano a buon fine sarà possibile inserire il nuovo voto sul Database. Inoltre, nel caso in cui il voto inserito non sia sufficiente, un servizio email dovrà inviare una notifica al genitore dello studente.

Immagine che contiene testo, diagramma, Parallelo, linea

Descrizione generata automaticamente

* *ModificaPagellaPreside*

Affinché il preside possa approvare una pagella per uno studente, è necessario effettuare controlli sull’esistenza dello studente: solo se lo studente esiste sarà possibile ottenere la pagella relativa e conoscere il suo stato. Se lo stato è “non approvata” il preside potrà aggiornarla e cambiare il suo stato in “approvata”.

Immagine che contiene testo, schermata, diagramma, Parallelo

Descrizione generata automaticamente

## Verifica della completezza dei requisiti

Per verificare la completezza dei requisiti, è possibile confrontare l’elenco dei requisiti funzionali (RF) e dei requisiti sui dati (RD) con i modelli Use Case Diagram e Class Diagram.

Legenda: UCD = Use Case Diagram, CD = Class Diagram

Si ottengono le seguenti corrispondenze:

* **RF01** è modellato nell’UCD con l’attore “Segreteria” e con il caso d’uso UC1
* **RF02** è modellato nell’UCD con l’attore “Docente” e con il caso d’uso UC3
* **RF03** è modellato nell’UCD con l’attore “Preside” e con il caso d’uso UC3
* **RF04** è modellato nell’UCD con l’attore “Docente” e con il caso d’uso UC4
* **RF05** è modellato nell’UCD con l’attore “Docente” e con il caso d’uso UC5
* **RF06** è modellato nell’UCD con il caso d’uso UC11, UC12
* **RF07** è modellato nell’UCD con il caso d’uso UC9
* **RF08** è modellato nell’UCD con il caso d’uso UC8
* **RF09** è modellato nell’UCD con il caso d’uso UC10
* **RF10** è modellato nell’UCD con il caso d’uso UC8
* **RF11** è modellato nell’UCD con l’attore “Docente” e con il caso d’uso UC6
* **RF12** è modellato nell’UCD con il caso d’uso UC8
* **RF13** è modellato nell’UCD con l’attore “Preside” e il caso d’uso UC7
* **RF14** è modellato nell’UCD con il caso d’uso UC6
* **RF15** è modellato nell’UCD con i casi d’uso UC11, UC13
* **RD01** è modellato nel CD con la classe “Utente”
* **RD02** è modellato nel CD con la classe “Istituto” e con la classe “Classe”
* **RD03** è modellato nel CD con gli attributi “sezione” e “anno” della classe “Classe”
* **RD04, RD05** è modellato nel CD con la classe “Registro Elettronico”
* **RD06** sono modellati nel CD con le classi “Docente” e “Materia Insegnata”
* **RD07** è modellato nel CD con gli attributi della classe “Studente”
* **RD08** è modellato nel CD con gli attributi della classe “Attività”
* **RD09** è modellato nel CD con l’attributo “data” della classe “Valutazione”
* **RD10** è modellato nel CD con l’attributo “stato” della classe “Pagella”

Dal momento che tutti i requisiti sono adeguatamente modellati negli Use Case Diagram e nei Class Diagram, la completezza dei requisiti è verificata.

1. **Stima dei costi**

Di seguito è riportata la stima dei costi secondo il metodo dei Punti Funzione, una metrica per la stima della quantità di funzionalità del software. Per farla si usano il numero di informazioni in entrata, in uscita e memorizzate dal sistema.

* Tabella di riferimento per le complessità di dati e transazioni

|  |  |  |  |
| --- | --- | --- | --- |
|  | **SEMPLICE** | **MEDIO** | **COMPLESSO** |
| **NILF** | **3** | **4** | **6** |
| **NEIF** | **4** | **5** | **7** |
| **NEI** | **3** | **4** | **6** |
| **NEO** | **7** | **10** | **15** |
| **NEQ** | **5** | **7** | **10** |

**AGGIUNGI VOTO**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **VALORE** | | **SEMPLICE** | | **MEDIO** | **COMPLESSO** | **TOT** |
| **NILF** | 1 | |  | | 4 |  | 4 |
| **NEIF** | 0 | |  | |  |  |  |
| **NEI** | 5 | | 3 | |  |  | 15 |
| **NEO** | 0 | |  | |  |  |  |
| **NEQ** | 4 | | 5 | |  |  | 20 |
| **UFP=** | | | **39** | |
| **LLOC/FP =** | | | 2.067 | |

NILF: Le valutazioni vengono gestiti internamente dal sistema, li identifichiamo come ILF. [1 medio]

NEI: Docente, Materia, Matricola, Voto Data [5 semplici]

NEQ: Check Docente, Check Materia, Check Matricola, Check Data [4 semplici]

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **FATTORI CORRETTIVI** | | | | | |
| **COMUNICAZIONE DATI** | | | | | 1 |
| **DISTRIBUZIONE ELABORAZIONE** | | | | | 0 |
| **PRESTAZIONI** | | | | | 1 |
| **UTILIZZO INTENSIVO CONFIGURAZIONE** | | | | | 0 |
| **FREQUENZA DELLE TRANSAZIONI** | | | | | 3 |
| **INSERIMENTO DATI INTERATTIVO** | | | | | 4 |
| **EFFICIENZA PER L’UTENTE FINALE** | | | | | 2 |
| **AGGIORNAMENTO INTERATTIVO** | | | | | 1 |
| **COMPLESSITA’ ELABORATIVA** | | | | | 2 |
| **RIUSABILITA’** | | | | | 3 |
| **FACILITA’ INSTALLAZIONE** | | | | | 0 |
| **FACILITA’ GESTIONE OPERATIVA** | | |  |  | 2 |
| **Molteplicità DI SITI** | | | | | 0 |
| **FACILITA’ DI MODIFICA** | | | | | 3 |
|  | | |  |  | 22 |
| **FP=** | **33,93** |
| **JAVA =** | **1.798** |

* COMUNICAZIONE DATI 1: I dati e le informazioni di controllo, utilizzati nell’applicazione, sono ricevuti o inviati attraverso l’interfaccia grafica. Però si tratta di un’applicazione che opera su PC stand alone quindi la comunicazione dati ha un’incidenza scarsa.
* DISTRIBUZIONE ELABORAZIONE 0: ininfluente
* PRESTAZIONI 1: I livelli di prestazione non hanno richiesto attenzione speciale durante lo sviluppo
* UTILIZZO INTENSIVO CONFIGURAZIONE 0: Non esistono vincoli di tipo operativo
* FREQUENZA DELLE TRANSAZIONI 3: Presenza di un alto volume di transazioni giornaliere
* INSERIMENTO DATI INTERATTIVO 4: La maggior parte dei dati e le funzioni di controllo sono presenti nell’interfaccia utente.
* EFFICIENZA PER L’UTENTE FINALE 2: Sono presenti almeno 4 o 5 elementi user-friendly come i vari tasti funzione, le stampe remote, la navigabilità tra le schermate, pop-up e messaggi di aiuto per l’utente
* AGGIORNAMENTO INTERATTIVO 1: L’applicazione prevede l’aggiornamento di pochi file di controllo. Dunque il volume degli aggiornamenti è contenuto.
* COMPLESSITÀ ELABORATIVA 2: Sono presenti meccanismi di elaborazione dei dati in input e di gestione delle eccezioni.
* RIUSABILITÀ 3: Più del 10% dei moduli prodotti considerano bisogni di più utenti. La documentazione rende più semplice il riutilizzo del codice.
* FACILITÀ INSTALLAZIONE 0: Non sono stati sviluppati particolari set-up per le installazioni.
* FACILITÀ GESTIONE OPERATIVA 1: La gestione operativa dell’applicazione non è considerata critica.
* MOLTEPLICITÀ DI SITI 0: L’applicazione non è progettata in modo specifico per essere installata su più siti.
* FACILITÀ DI MODIFICA 3: L’applicazione è progettata in modo tale da minimizzare gli impatti di modifiche. I dati di controllo sono inseriti in tabelle che sono gestite dall’utente e le variazioni sono soggette a controllo.

**VISUALIZZA VOTI**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **VALORE** | | **SEMPLICE** | | **MEDIO** | **COMPLESSO** | **TOT** |
| **NILF** | 1 | |  | | 4 |  | 4 |
| **NEIF** | 0 | |  | |  |  |  |
| **NEI** | 1 | | 3 | |  |  | 3 |
| **NEO** | 0 | |  | |  |  |  |
| **NEQ** | 2 | | 5 | | 7 |  | 12 |
| **UFP=** | | | **19** | |
| **LLOC/FP =** | | | 1.007 | |

NILF: Le valutazioni vengono gestiti internamente dal sistema, li identifichiamo come ILF. [1 medio]

NEI: UsernameGenitore, [1 semplice]

NEQ: Check Genitore, Visualizzazione dei voti [1 semplice, 1 complesso]

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **FATTORI CORRETTIVI** | | | | | |
| **COMUNICAZIONE DATI** | | | | | 1 |
| **DISTRIBUZIONE ELABORAZIONE** | | | | | 0 |
| **PRESTAZIONI** | | | | | 1 |
| **UTILIZZO INTENSIVO CONFIGURAZIONE** | | | | | 0 |
| **FREQUENZA DELLE TRANSAZIONI** | | | | | 3 |
| **INSERIMENTO DATI INTERATTIVO** | | | | | 1 |
| **EFFICIENZA PER L’UTENTE FINALE** | | | | | 2 |
| **AGGIORNAMENTO INTERATTIVO** | | | | | 0 |
| **COMPLESSITA’ ELABORATIVA** | | | | | 2 |
| **RIUSABILITA’** | | | | | 3 |
| **FACILITA’ INSTALLAZIONE** | | | | | 0 |
| **FACILITA’ GESTIONE OPERATIVA** | | |  |  | 1 |
| **Molteplicità DI SITI** | | | | | 0 |
| **FACILITA’ DI MODIFICA** | | | | | 2 |
|  | | |  |  | 22 |
| **FP=** | **15,39** |
| **JAVA =** | **816** |

* COMUNICAZIONE DATI 1: I dati e le informazioni di controllo, utilizzati nell’applicazione, sono ricevuti o inviati attraverso l’interfaccia grafica. Però si tratta di un’applicazione che opera su PC stand alone quindi la comunicazione dati ha un’incidenza scarsa.
* DISTRIBUZIONE ELABORAZIONE 0: ininfluente
* PRESTAZIONI 1: I livelli di prestazione non hanno richiesto attenzione speciale durante lo sviluppo
* UTILIZZO INTENSIVO CONFIGURAZIONE 0: Non esistono vincoli di tipo operativo
* FREQUENZA DELLE TRANSAZIONI 3: Presenza di un alto volume di transazioni giornaliere
* INSERIMENTO DATI INTERATTIVO 1: Gli inserimenti tramite interfaccia utente sono contenuti e riguardano solo funzioni di controllo.
* EFFICIENZA PER L’UTENTE FINALE 2: Sono presenti almeno 4 o 5 elementi user-friendly come i vari tasti funzione, le stampe remote, la navigabilità tra le schermate, pop-up e messaggi di aiuto per l’utente
* AGGIORNAMENTO INTERATTIVO 0: L’applicazione non prevede l’aggiornamento di file di controllo.
* COMPLESSITÀ ELABORATIVA 1: Sono presenti meccanismi di elaborazione dei dati in output e di gestione delle eccezioni.
* RIUSABILITÀ 3: Più del 10% dei moduli prodotti considerano bisogni di più utenti. La documentazione rende più semplice il riutilizzo del codice.
* FACILITÀ INSTALLAZIONE 0: Non sono stati sviluppati particolari set-up per le installazioni.
* FACILITÀ GESTIONE OPERATIVA 1: La gestione operativa dell’applicazione non è considerata critica.
* MOLTEPLICITÀ DI SITI 0: L’applicazione non è progettata in modo specifico per essere installata su più siti.
* FACILITÀ DI MODIFICA 2: L’applicazione è progettata in modo tale da minimizzare gli impatti di modifiche.

**REGISTRA UTENTE**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **VALORE** | | **SEMPLICE** | | **MEDIO** | **COMPLESSO** | **TOT** |
| **NILF** | 5 | | 3 | | 10 |  | 36 |
| **NEIF** | 0 | |  | |  |  |  |
| **NEI** | 13 | | 3 | |  |  | 39 |
| **NEO** | 0 | |  | |  |  |  |
| **NEQ** | 7 | | 5 | | 7 |  | 41 |
| **UFP=** | | | **116** | |
| **LLOC/FP =** | | | **6.147** | |

NILF: Docente, Studente e Genitore vengono creati dal sistema, li identifichiamo come ILF. Materia e Classe vengono aggiornate. [3 medie e 2 semplici]

NEI: Ruolo, Nome, Cognome, Data di nascita, Codice Fiscale, Comune di residenza, Email, Numero di cellulare, Username, Password, Materia, Figlio, Classe [13 semplici]

NEQ: Check Username, Check Materia, Check Studente, Check Classe, Lista Materie, Lista Matricola, Lista Classe [4 semplice, 3 medi]

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **FATTORI CORRETTIVI** | | | | | |
| **COMUNICAZIONE DATI** | | | | | 1 |
| **DISTRIBUZIONE ELABORAZIONE** | | | | | 0 |
| **PRESTAZIONI** | | | | | 1 |
| **UTILIZZO INTENSIVO CONFIGURAZIONE** | | | | | 0 |
| **FREQUENZA DELLE TRANSAZIONI** | | | | | 1 |
| **INSERIMENTO DATI INTERATTIVO** | | | | | 4 |
| **EFFICIENZA PER L’UTENTE FINALE** | | | | | 2 |
| **AGGIORNAMENTO INTERATTIVO** | | | | | 2 |
| **COMPLESSITA’ ELABORATIVA** | | | | | 3 |
| **RIUSABILITA’** | | | | | 3 |
| **FACILITA’ INSTALLAZIONE** | | | | | 0 |
| **FACILITA’ GESTIONE OPERATIVA** | | |  |  | 1 |
| **Molteplicità DI SITI** | | | | | 0 |
| **FACILITA’ DI MODIFICA** | | | | | 3 |
|  | | |  |  | **21** |
| **FP=** | **99,76** |
| **JAVA =** | **5.287** |

* COMUNICAZIONE DATI 1: I dati e le informazioni di controllo, utilizzati nell’applicazione, sono ricevuti o inviati attraverso l’interfaccia grafica. Però si tratta di un’applicazione che opera su PC stand alone quindi la comunicazione dati ha un’incidenza scarsa.
* DISTRIBUZIONE ELABORAZIONE 0: ininfluente
* PRESTAZIONI 1: I livelli di prestazione non hanno richiesto attenzione speciale durante lo sviluppo
* UTILIZZO INTENSIVO CONFIGURAZIONE 0: Non esistono vincoli di tipo operativo
* FREQUENZA DELLE TRANSAZIONI 1: Presenza di picchi di carico di transazioni periodicamente.
* INSERIMENTO DATI INTERATTIVO 4: La maggior parte dei dati e le funzioni di controllo sono presenti nell’interfaccia utente.
* EFFICIENZA PER L’UTENTE FINALE 2: Sono presenti almeno 4 o 5 elementi user-friendly come i vari tasti funzione, le stampe remote, la navigabilità tra le schermate, pop-up e messaggi di aiuto per l’utente
* AGGIORNAMENTO INTERATTIVO 2: L’applicazione prevede l’aggiornamento di alcuni file di controllo.
* COMPLESSITÀ ELABORATIVA 3: Sono presenti molti meccanismi di elaborazione dei dati in input e di gestione delle eccezioni.
* RIUSABILITÀ 3: Più del 10% dei moduli prodotti considerano bisogni di più utenti. La documentazione rende più semplice il riutilizzo del codice.
* FACILITÀ INSTALLAZIONE 0: Non sono stati sviluppati particolari set-up per le installazioni.
* FACILITÀ GESTIONE OPERATIVA 1: La gestione operativa dell’applicazione non è considerata critica.
* MOLTEPLICITÀ DI SITI 0: L’applicazione non è progettata in modo specifico per essere installata su più siti.
* FACILITÀ DI MODIFICA 3: L’applicazione è progettata in modo tale da minimizzare gli impatti di modifiche. I dati di controllo sono inserite in tabelle che sono gestite dall’utente e le variazioni sono soggette a controllo.

# Piano di test funzionale

Per il piano di test funzionale si è utilizzato il metodo del Category-Partition Testing. Questa metodologia di test è basata sulla suddivisione delle possibili combinazioni di input in diverse categorie. Ciò ci consente di ridurre il numero di casi di test necessari per coprire tutte le possibili combinazioni, garantendo comunque una copertura adeguata per i vari scenari e casi limite.

**PIANO DI TEST UTILIZZANDO IL METODO DEL *CATEGORY-PARTITION TESTING* PER LA FUNZIONALITÀ “*RegistraUtente*”.**

|  |  |  |  |
| --- | --- | --- | --- |
| **NOME** | **COGNOME** | **DATA DI NASCITA** | **CODICE FISCALE** |
| * Stringa di caratteri che inizia con una maiuscola di lunghezza > 0 e <= 100 * Stringa di caratteri che inizia con una minuscola [ERROR] * Stringa vuota [ERROR] * Stringa di lunghezza > 100 [ERROR] | * Stringa di caratteri che inizia con una maiuscola di lunghezza > 0 e <= 100 * Stringa di caratteri che inizia con una minuscola [ERROR] * Stringa vuota [ERROR] * Stringa di lunghezza > 100 [ERROR] | * Data con formato valido(gg/mm/aaaa) * Data con formato non valido [ERROR] | * Stringa alfanumerica di lunghezza = 16 con caratteri maiuscoli * Stringa alfanumerica di lunghezza != 16 [ERROR] * Stringa composta da simboli non validi [ERROR] |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **COMUNE DI RESIDENZA** | **EMAIL** | **NUMERO DI CELLULARE** | **USERNAME** | **PASSWORD** | **RUOLO** |
| * Stringa di caratteri di lunghezza > 0 e <=50 * Stringa vuota [ERROR] * Stringa di lunghezza > 50 [ERROR] * Stringa composta da simboli non validi [ERROR] | * Stringa in cui è presente il simbolo @ * Stringa in cui non è presente il simbolo @ [ERROR] | * Stringa di lunghezza >0 e <15 * Stringa vuota [ERROR] * Stringa di lunghezza > 15[ERROR] | * Stringa alfanumerica di lunghezza >0 e <=20 * Stringa vuota [ERROR] * Stringa di lunghezza > 20 [ERROR] | * Stringa alfanumerica di lunghezza > 0 e <= 50 * Stringa vuota [ERROR] * Stringa di lunghezza > 50 [ERROR] | * Stringa = “Docente” * Stringa = “Genitore” * Stringa = “Alunno” * Stringa != {Docente, Genitore, Alunno} [ERROR] |

Il numero di test da effettuarsi senza particolari vincoli è:

Introduciamo i vincoli [ERROR] .

Il numero di test da eseguire per testare singolarmente i vincoli è 20 (3 per Nome, 3 per Cognome, 1 per DataNascita, 2 per CodiceFiscale, 3 per ComuneResidenza, 1 per Email, 2 per NumeroCellulare, 2 per Username, 2 per Password, 1 per Ruolo)

Il numero di test risultante è: (1\*1\*1\*1\*1\*1\*1\*1\*1\*3) + 20 = 23.

**TEST SUITE**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Descrizione** | **Classi di equivalenza coperte** | **Pre-condizioni** | **Input** | **Output Attesi** | **Post-condizioni Attese** |
| 1 | Tutti input validi e Ruolo = “Docente” | Nome valido, Cognome valido, DataNascita valida, CodiceFiscale, ComuneResidenza valido, Email valida, NumeroCellulare valido, Username valido, Password valida, Ruolo = “Docente” valido | Username non registrato | {Nome: “Luigi”, Cognome: “Bianchi”, DataNascita: “30/11/1975”, CF: “BNCLGU75S30H703D”,  ComuneResidenza: “Salerno”, Email: [luigibianchi75@gmail.com](mailto:luigibianchi75@gmail.com),  NumeroCellulare: “338763435”, Username: “l.bianchi30”, Password: “BianchiPass3@”, Ruolo: “Docente”} | “Utente Registrato” | Docente aggiunto al Sistema con I relativi privilegi e le relative materie insegnata. |
| 2 | Tutti input validi e Ruolo = “Alunno” | Nome valido, Cognome valido, DataNascita valida, CodiceFiscale, ComuneResidenza valido, Email valida, NumeroCellulare valido, Username valido, Password valida, Ruolo = “Alunno” valido | Username non registrato | {Nome: “Mario”, Cognome: “Rossi”, DataNascita: “20/10/2001”, CF: “RSSMRA01R20F839P”,  ComuneResidenza: “Napoli”, Email: [mariorossi2001@gmail.com](mailto:mariorossi2001@gmail.com),  NumeroCellulare: “338765432”, Username: “m.rossi20”, Password: “ScuolaMario20@”, Ruolo: “Alunno”} | “Utente Registrato” | Alunno aggiunto al Sistema con I relativi privilegi a cui viene assegnata una matricola e viene specificata la classe di appartenenza. |
| 3 | Tutti input validi e Ruolo = “Genitore” | Nome valido, Cognome valido, DataNascita valida, CodiceFiscale, ComuneResidenza valido, Email valida, NumeroCellulare valido, Username valido, Password valida, Ruolo = “Genitore” valido | Username non registrato | {Nome: “Carla”, Cognome: “Viola”, DataNascita: “13/06/1980”, CF: “CRLVLI80H53F839Z”,  ComuneResidenza: “Napoli”, Email: [carlaviola80@gmail.com](mailto:carlaviola80@gmail.com),  NumeroCellulare: “327673435”, Username: “c.viola13”, Password: “IstitutoCarla80”, Ruolo: “Genitore”} | “Utente Registrato” | Genitore aggiunto al Sistema con I relativi privilegi e viene specificata la matricola dello studente di cui è genitore. |
| 4 | Nome stringa di caratteri che inizia con una minuscola | Nome stringa che inizia con minuscola [ERROR}, Cognome valido, DataNascita valida, CodiceFiscale, ComuneResidenza valido, Email valida, NumeroCellulare valido, Username valido, Password valida, Ruolo = “Genitore” valido |  | {Nome: “carla”, Cognome: “Viola”, DataNascita: “13/06/1980”, CF: “CRLVLI80H53F839Z”,  ComuneResidenza: “Napoli”, Email: [carlaviola80@gmail.com](mailto:carlaviola80@gmail.com),  NumeroCellulare: “327673435”, Username: “c.viola13”, Password: “IstitutoCarla”, Ruolo: “Genitore”} | “Il nome deve iniziare con una lettera maiuscola” |  |
| 5 | Nome stringa vuota | Nome stringa vuota [ERROR}, Cognome valido, DataNascita valida, CodiceFiscale, ComuneResidenza valido, Email valida, NumeroCellulare valido, Username valido, Password valida, Ruolo = “Genitore” valido |  | {Nome: “ ”, Cognome: “Viola”, DataNascita: “13/06/1980”, CF: “CRLVLI80H53F839Z”,  ComuneResidenza: “Napoli”, Email: [carlaviola80@gmail.com](mailto:carlaviola80@gmail.com),  NumeroCellulare: “327673435”, Username: “c.viola13”, Password: “IstitutoCarla”, Ruolo: “Genitore”} | “Nome non valido” |  |
| 6 | Nome stringa di lunghezza >100 | Nome stringa > 100 [ERROR}, Cognome valido, DataNascita valida, CodiceFiscale, ComuneResidenza valido, Email valida, NumeroCellulare valido, Username valido, Password valida, Ruolo = “Genitore” valido |  | {Nome: “ Carla Carla Carla Carla Carla Carla Carla Carla Carla Carla Carla Carla Carla Carla Carla Carla Carla Carla Carla Carla Carla”, Cognome: “Viola”, DataNascita: “13/06/1980”, CF: “CRLVLI80H53F839Z”,  ComuneResidenza: “Napoli”, Email: [carlaviola80@gmail.com](mailto:carlaviola80@gmail.com),  NumeroCellulare: “327673435”, Username: “c.viola13”, Password: “IstitutoCarla”, Ruolo: “Genitore”} | “Nome non valido” |  |
| 7 | Cognome stringa di caratteri che inizia con una minuscola | Nome valido, Cognome stringa che inizia con minuscola [ERROR}, DataNascita valida, CodiceFiscale, ComuneResidenza valido, Email valida, NumeroCellulare valido, Username valido, Password valida, Ruolo = “Genitore” valido |  | {Nome: “Carla”, Cognome: “viola”, DataNascita: “13/06/1980”, CF: “CRLVLI80H53F839Z”,  ComuneResidenza: “Napoli”, Email: [carlaviola80@gmail.com](mailto:carlaviola80@gmail.com),  NumeroCellulare: “327673435”, Username: “c.viola13”, Password: “IstitutoCarla”, Ruolo: “Genitore”} | “Il cognome deve iniziare con una maiuscola” |  |
| 8 | Cognome stringa vuota | Nome valido, Cognome stringa vuota [ERROR}, DataNascita valida, CodiceFiscale, ComuneResidenza valido, Email valida, NumeroCellulare valido, Username valido, Password valida, Ruolo = “Genitore” valido |  | {Nome: “Carla”, Cognome: “ ”, DataNascita: “13/06/1980”, CF: “CRLVLI80H53F839Z”,  ComuneResidenza: “Napoli”, Email: [carlaviola80@gmail.com](mailto:carlaviola80@gmail.com),  NumeroCellulare: “327673435”, Username: “c.viola13”, Password: “IstitutoCarla”, Ruolo: “Genitore”} | “Cognome non valido” |  |
| 9 | Cognome stringa di lunghezza >100 | Nome valido, Cognome stringa > 100 [ERROR}, DataNascita valida, CodiceFiscale, ComuneResidenza valido, Email valida, NumeroCellulare valido, Username valido, Password valida, Ruolo = “Genitore” valido |  | {Nome: “Carla”, Cognome: “Viola Viola Viola Viola Viola Viola Viola Viola Viola Viola Viola Viola Viola Viola Viola Viola Viola Viola Viola Viola Viola”, DataNascita: “13/06/1980”, CF: “CRLVLI80H53F839Z”,  ComuneResidenza: “Napoli”, Email: [carlaviola80@gmail.com](mailto:carlaviola80@gmail.com),  NumeroCellulare: “327673435”, Username: “c.viola13”, Password: “IstitutoCarla”, Ruolo: “Genitore”} | “Cognome non valido” |  |
| 10 | Data con formato non valido | Nome valido, Cognome valido ,DataNascita formato non valido [ERROR], CodiceFiscale, ComuneResidenza valido, Email valida, NumeroCellulare valido, Username valido, Password valida, Ruolo = “Genitore” valido |  | {Nome: “Carla”, Cognome: “Viola”, DataNascita: “130/060/19800”, CF: “CRLVLI80H53F839Z”,  ComuneResidenza: “Napoli”, Email: [carlaviola80@gmail.com](mailto:carlaviola80@gmail.com),  NumeroCellulare: “327673435”, Username: “c.viola13”, Password: “IstitutoCarla”, Ruolo: “Genitore”} | “Data non valida” |  |
| 11 | Codice Fiscale stringa alfanumerica di lunghezza != 16 | Nome valido, Cognome valido ,DataNascita valida, CodiceFiscale stringa != 16 [ERROR], ComuneResidenza valido, Email valida, NumeroCellulare valido, Username valido, Password valida, Ruolo = “Genitore” valido |  | {Nome: “Carla”, Cognome: “Viola”, DataNascita: “13/06/1980”, CF: “CRLVLI80H53F839Z910”,  ComuneResidenza: “Napoli”, Email: [carlaviola80@gmail.com](mailto:carlaviola80@gmail.com),  NumeroCellulare: “327673435”, Username: “c.viola13”, Password: “IstitutoCarla”, Ruolo: “Genitore”} | “Codice fiscale non valido” |  |
| 12 | Codice Fiscale composta da simboli non validi | Nome valido, Cognome valido ,DataNascita valida, CodiceFiscale stringa != 16 [ERROR], ComuneResidenza valido, Email valida, NumeroCellulare valido, Username valido, Password valida, Ruolo = “Genitore” valido |  | {Nome: “Carla”, Cognome: “Viola”, DataNascita: “13/06/1980”, CF: “CR%L&80/5&F8$9ZG”,  ComuneResidenza: “Napoli”, Email: [carlaviola80@gmail.com](mailto:carlaviola80@gmail.com),  NumeroCellulare: “327673435”, Username: “c.viola13”, Password: “IstitutoCarla”, Ruolo: “Genitore”} | “Codice fiscale non valido” |  |
| 13 | Comune di residenza stringa vuota | Nome valido, Cognome valido, DataNascita valida, CodiceFiscale, ComuneResidenza stringa vuota [ERROR], Email valida, NumeroCellulare valido, Username valido, Password valida, Ruolo = “Genitore” valido |  | {Nome: “Carla”, Cognome: “Viola”, DataNascita: “13/06/1980”, CF: “CRLVLI80H53F839Z”,  ComuneResidenza: “ ”, Email: [carlaviola80@gmail.com](mailto:carlaviola80@gmail.com),  NumeroCellulare: “327673435”, Username: “c.viola13”, Password: “IstitutoCarla”, Ruolo: “Genitore”} | “Comune di residenza non valido” |  |
| 14 | Comune di Residenza stringa di lunghezza > 50 | Nome valido, Cognome valido, DataNascita valida, CodiceFiscale, ComuneResidenza stringa > 50 [ERROR], Email valida, NumeroCellulare valido, Username valido, Password valida, Ruolo = “Genitore” valido |  | {Nome: “Carla”, Cognome: “Viola”, DataNascita: “13/06/1980”, CF: “CRLVLI80H53F839Z”,  ComuneResidenza: “Napoli Napoli Napoli Napoli Napoli Napoli Napoli Napoli Napoli”, Email: [carlaviola80@gmail.com](mailto:carlaviola80@gmail.com),  NumeroCellulare: “327673435”, Username: “c.viola13”, Password: “IstitutoCarla”, Ruolo: “Genitore”} | “Comune di residenza non valido” |  |
| 15 | Comune di Residenza stringa composta da simboli non validi | Nome valido, Cognome valido, DataNascita valida, CodiceFiscale, ComuneResidenza stringa con simboli non validi [ERROR], Email valida, NumeroCellulare valido, Username valido, Password valida, Ruolo = “Genitore” valido |  | {Nome: “Carla”, Cognome: “Viola”, DataNascita: “13/06/1980”, CF: “CRLVLI80H53F839Z”,  ComuneResidenza: “N@p0li”, Email: [carlaviola80@gmail.com](mailto:carlaviola80@gmail.com),  NumeroCellulare: “327673435”, Username: “c.viola13”, Password: “IstitutoCarla”, Ruolo: “Genitore”} | “Comune non valido” |  |
| 16 | Email stringa in cui non è presente il simbolo @ | Nome valido, Cognome valido, DataNascita valida, CodiceFiscale, ComuneResidenza valido, Email stringa in cui non è presente @ [ERROR], NumeroCellulare valido, Username valido, Password valida, Ruolo = “Genitore” valido |  | {Nome: “Carla”, Cognome: “Viola”, DataNascita: “13/06/1980”, CF: “CRLVLI80H53F839Z”,  ComuneResidenza: “Napoli”, Email: [carlaviola80.gmail.com](mailto:carlaviola80@gmail.com),  NumeroCellulare: “327673435”, Username: “c.viola13”, Password: “IstitutoCarla”, Ruolo: “Genitore”} | “Email non valida” |  |
| 17 | Numero Cellulare Stringa vuota | Nome valido, Cognome valido, DataNascita valida, CodiceFiscale, ComuneResidenza valido, Email valida, NumeroCellulare stringa vuota [ERROR], Username valido, Password valida, Ruolo = “Genitore” valido |  | {Nome: “Carla”, Cognome: “Viola”, DataNascita: “13/06/1980”, CF: “CRLVLI80H53F839Z”,  ComuneResidenza: “Napoli”, Email: [carlaviola80@gmail.com](mailto:carlaviola80@gmail.com),  NumeroCellulare: “ ”, Username: “c.viola13”, Password: “IstitutoCarla”, Ruolo: “Genitore”} | “Numero di cellulare non valido” |  |
| 18 | Numero Cellulare stringa di lunghezza > 15 | Nome valido, Cognome valido, DataNascita valida, CodiceFiscale, ComuneResidenza valido, Email valida, NumeroCellulare stringa > 15 [ERROR], Username valido, Password valida, Ruolo = “Genitore” valido |  | {Nome: “Carla”, Cognome: “Viola”, DataNascita: “13/06/1980”, CF: “CRLVLI80H53F839Z”,  ComuneResidenza: “Napoli”, Email: [carlaviola80@gmail.com](mailto:carlaviola80@gmail.com),  NumeroCellulare: “+39 33920182038402810 ”, Username: “c.viola13”, Password: “IstitutoCarla”, Ruolo: “Genitore”} | “Numero di cellulare non valido” |  |
| 19 | Username stringa vuota | Nome valido, Cognome valido, DataNascita valida, CodiceFiscale, ComuneResidenza valido, Email valida, NumeroCellulare valido, Username stringa vuota [ERROR], Password valida, Ruolo = “Genitore” valido |  | {Nome: “Carla”, Cognome: “Viola”, DataNascita: “13/06/1980”, CF: “CRLVLI80H53F839Z”,  ComuneResidenza: “Napoli”, Email: [carlaviola80@gmail.com](mailto:carlaviola80@gmail.com),  NumeroCellulare: “327673435”, Username: “ ”, Password: “IstitutoCarla”, Ruolo: “Genitore”} | “Username non valido” |  |
| 20 | Username stringa di lunghezza > 20 | Nome valido, Cognome valido, DataNascita valida, CodiceFiscale, ComuneResidenza valido, Email valida, NumeroCellulare valido, Username stringa > 20 [ERROR], Password valida, Ruolo = “Genitore” valido |  | {Nome: “Carla”, Cognome: “Viola”, DataNascita: “13/06/1980”, CF: “CRLVLI80H53F839Z”,  ComuneResidenza: “Napoli”, Email: [carlaviola80@gmail.com](mailto:carlaviola80@gmail.com),  NumeroCellulare: “327673435”, Username: “carla.viola13278738193037”, Password: “IstitutoCarla”, Ruolo: “Genitore”} | “Username non valido” |  |
| 21 | Password stringa vuota | Nome valido, Cognome valido, DataNascita valida, CodiceFiscale, ComuneResidenza valido, Email valida, NumeroCellulare valido, Username valido, Password stringa vuota [ERROR], Ruolo = “Genitore” valido |  | {Nome: “Carla”, Cognome: “Viola”, DataNascita: “13/06/1980”, CF: “CRLVLI80H53F839Z”,  ComuneResidenza: “Napoli”, Email: [carlaviola80@gmail.com](mailto:carlaviola80@gmail.com),  NumeroCellulare: “327673435”, Username: “c.viola13”, Password: “ ”, Ruolo: “Genitore”} | “Password non valida” |  |
| 22 | Password stringa di lunghezza > 50 | Nome valido, Cognome valido, DataNascita valida, CodiceFiscale, ComuneResidenza valido, Email valida, NumeroCellulare valido, Username valido, Password stringa > 50[ERROR], Ruolo = “Genitore” valido |  | {Nome: “Carla”, Cognome: “Viola”, DataNascita: “13/06/1980”, CF: “CRLVLI80H53F839Z”,  ComuneResidenza: “Napoli”, Email: [carlaviola80@gmail.com](mailto:carlaviola80@gmail.com),  NumeroCellulare: “327673435”, Username: “c.viola13”, Password: “IstitutoCarlaIstitutoCarlaIstitutoCarlaIstitutoCarlaIstitutoCarla”, Ruolo: “Genitore”} | “Password non valida” |  |
| 23 | Ruolo stringa != {Docente, Genitore, Alunno} | Nome valido, Cognome valido, DataNascita valida, CodiceFiscale, ComuneResidenza valido, Email valida, NumeroCellulare valido, Username valido, Password valida, Ruolo stringa != {Docente, Genitore, Alunno} [ERROR] |  | {Nome: “Carla”, Cognome: “Viola”, DataNascita: “13/06/1980”, CF: “CRLVLI80H53F839Z”,  ComuneResidenza: “Napoli”, Email: [carlaviola80@gmail.com](mailto:carlaviola80@gmail.com),  NumeroCellulare: “327673435”, Username: “c.viola13”, Password: “IstitutoCarla”, Ruolo: “Madre”} | “Ruolo non valido; deve essere o uno Studente o un Genitore o un Docente” |  |

**PIANO DI TEST UTILIZZANDO IL METODO DEL *CATEGORY-PARTITION TESTING* PER LA FUNZIONALITÀ “*InserisciClasse*”.**

|  |  |  |
| --- | --- | --- |
| **ANNO** | **SEZIONE** | **ANNO SCOLASTICO** |
| * Stringa = “I” * Stringa = “II” * Stringa = “III” * Stringa = “IV” * Stringa = “V” * Stringa != dall’insieme di elementi {“I”,”II”,”III”,”IV”,”V”} [ERROR] | * Stringa composta da un carattere alfabetico maiuscolo * Stringa di caratteri di lunghezza != 1 [ERROR] * Stringa composta da un carattere alfabetico minuscolo [ERROR] * Stringa composta da un carattere numerico [ERROR] * Stringa che contiene simboli non validi [ERROR] | * Intero di lunghezza = 4 i cui valori sono >= 1900 e <= 2100 * Intero di lunghezza < 4 [ERROR] * Intero di lunghezza > 4 [ERROR] |

Il numero di test da effettuarsi senza particolari vincoli è:

Introduciamo i vincoli [ERROR] .

Il numero di test da eseguire per testare singolarmente i vincoli è 7 (1 per Anno, 4 per Sezione, 2 per AnnoScolastico)

Il numero di test risultante è: (5\*1\*1) + 7 = 12

**TEST SUITE**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Descrizione** | **Classi di equivalenza coperte** | **Pre-condizioni** | **Input** | **Output Attesi** | **Post-condizioni Attese** |
| 1 | Tutti input validi con Anno = “I” | Anno = “I” valido  Sezione valida  AnnoScolastico valido |  | {Anno: “I”,  Sezione: “A”,  AnnoScolastico: “2022”  } | “CLASSE CREATA”  e aggiunta all’anno scolastico di un trattino con l’anno consecutivo a quello inserito | Classe Creata |
| 2 | Tutti input validi con Anno = “II” | Anno = “II” valido  Sezione valida  AnnoScolastico valido |  | {Anno: “II”,  Sezione: “A”,  AnnoScolastico: “2022”  } | “CLASSE CREATA”  e aggiunta all’anno scolastico di un trattino con l’anno consecutivo a quello inserito | Classe Creata |
| 3 | Tutti input validi con Anno = “III” | Anno = “III” valido  Sezione valida  AnnoScolastico valido |  | {Anno: “III”,  Sezione: “A”,  AnnoScolastico: “2022”  } | “CLASSE CREATA”  e aggiunta all’anno scolastico di un trattino con l’anno consecutivo a quello inserito | Classe Creata |
| 4 | Tutti input validi con Anno = “IV” | Anno = “IV” valido  Sezione valida  AnnoScolastico valido |  | {Anno: “IV”,  Sezione: “A”,  AnnoScolastico: “2022”  } | “CLASSE CREATA”  e aggiunta all’anno scolastico di un trattino con l’anno consecutivo a quello inserito | Classe Creata |
| 5 | Tutti input validi con Anno = “V” | Anno = “V” valido  Sezione valida  AnnoScolastico valido |  | {Anno: “V”,  Sezione: “A”,  AnnoScolastico: “2022”  } | “CLASSE CREATA”  e aggiunta all’anno scolastico di un trattino con l’anno consecutivo a quello inserito | Classe Creata |
| 6 | Anno stringa non appartenente all’insieme degli anni validi | Anno stringa != dall’insieme di elementi {“I”,”II”,”III”,”IV”,”V”} [ERROR]  Sezione valida, AnnoScolastico validi |  | {Anno: “IIII”,  Sezione: “A”,  AnnoScolastico: “2022”  } | “Anno non valido!” |  |
| 7 | Sezione stringa di caratteri di lunghezza != 1 | Anno valido, Sezione stringa > 1 [ERROR], AnnoScolastico validi |  | {Anno: “V”,  Sezione: “AAA”,  AnnoScolastico: “2022”  } | “Sezione della classe troppo lunga!” |  |
| 8 | Sezione stringa composta da un carattere alfabetico minuscolo | Anno valido, Sezione con carattere minuscolo [ERROR], AnnoScolastico validi |  | {Anno: “V”,  Sezione: “a”,  AnnoScolastico: “2022”  } | “Sezione della classe minuscola, non valida!” |  |
| 9 | Sezione stringa composta da un carattere numerico | Anno valido, Sezione con carattere numerico[ERROR], AnnoScolastico validi |  | {Anno: “V”,  Sezione: “5”,  AnnoScolastico: “2022”  } | “Sezione della classe numerica, non valida!” |  |
| 10 | Sezione stringa che contiene simboli non validi | Anno valido, Sezione con simboli [ERROR], AnnoScolastico validi |  | {Anno: “V”,  Sezione: “&”,  AnnoScolastico: “2022”  } | “Sezione della classe simbolica, non valida!” |  |
| 11 | AnnoScolastico intero di lunghezza < 4 | Anno valido, Sezione valida, AnnoScolastico < 4 [ERROR] |  | {Anno: “V”,  Sezione: “A”,  AnnoScolastico: “202”  } | “Anno Scolastico troppo corto!” |  |
| 12 | AnnoScolastico intero di lunghezza > 4 | Anno valido, Sezione valida, AnnoScolastico > 4 [ERROR] |  | {Anno: “V”,  Sezione: “A”,  AnnoScolastico: “20222”  } | “Anno Scolastico troppo lungo!” |  |

**PIANO DI TEST UTILIZZANDO IL METODO DEL *CATEGORY-PARTITION TESTING* PER LA FUNZIONALITÀ “*RiportaAttività*”.**

|  |  |
| --- | --- |
| **DATA** | **DESCRIZIONE** |
| * Data con formato valido(gg/mm/aaaa) * Data con formato non valido [ERROR] | * Stringa alfanumerica di lunghezza <= 100 * Stringa di lunghezza >100 [ERROR] |

Il numero di test da effettuarsi senza particolari vincoli è:

Introduciamo i vincoli [ERROR].

Il numero di test da eseguire per testare singolarmente i vincoli è 2 (1 per Data,1 per Descrizione).

Il numero di test risultante è: (1\*1) + 2 = 3

**TEST SUITE**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Descrizione** | **Classi di equivalenza coperte** | **Pre-condizioni** | **Input** | **Output Attesi** | **Post-condizioni Attese** |
| 1 | Tutti input validi | Data valida, Descrizione valida | È stata selezionata una classe esistente, la data è inclusa nel quadrimestre corrente | {Data: “20/10/2022”, Descrizione: “Interrogazione di Italiano”} | “ATTIVITÁ CREATA” | Viene istanziata l’attività sul registro elettronico |
| 2 | Data formato non valido | Data formato non valido [ERROR], Descrizione valida |  | {Data: “200/100/20222”, Descrizione: “Interrogazione di Italiano”} | “Data non valida!” |  |
| 3 | Descrizione stringa > 100 | Data valida, Descrizione stringa > 100 caratteri [ERROR] |  | {Data: “200/100/20222”, Descrizione: “Interrogazione di Italiano. Interrogazione di Italiano. Interrogazione di Italiano. Interrogazione di Italiano Interrogazione di Italiano Interrogazione di Italiano”} | “Descrizione troppo lunga, non valida!” |  |

**PIANO DI TEST UTILIZZANDO IL METODO DEL *CATEGORY-PARTITION TESTING* PER LA FUNZIONALITÀ “*AggiungiVoto*”.**

|  |  |  |
| --- | --- | --- |
| **DATA** | **ALUNNO** | **VALUTAZIONE** |
| * Data con formato valido(gg/mm/aaaa) * Data con formato non valido [ERROR] | * Intero di lunghezza = 6 * Intero di lunghezza != 6 [ERROR] | * Float con valori >=1 e <=10 * Float con valori < 1 [ERROR] * Float con valori >10 [ERROR] |

Il numero di test da effettuarsi senza particolari vincoli è:

Introduciamo i vincoli [ERROR] .

Il numero di test da eseguire per testare singolarmente i vincoli è 4 (1 per Data, 1 per Alunno, 2 per Valutazione).

Il numero di test risultante è: (1\*1\*1) + 4 = 5

**TEST SUITE**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Descrizione** | **Classi di equivalenza coperte** | **Pre-condizioni** | **Input** | **Output Attesi** | **Post-condizioni Attese** |
| 1 | Tutti input validi | Data valida, Alunno valido, Valutazione valida. | È stata selezionata una classe esistente, la data è inclusa nel quadrimestre corrente, l’alunno esiste e fa parte della classe selezionata. | { Classe: “III A”, Data: “20/10/2022”, Alunno: 000001, Valutazione: 4.5} | “Valutazione inserita” | La valutazione viene inserita nel registro elettronico e a causa della valutazione negativa il sistema invia una email al genitore dell’alunno. |
| 2 | Data formato non valido | Data formato non valido [ERROR], Alunno valido, Valutazione valida. |  | {Classe: “III A”, Data: “200/100/20222”, Alunno: 000001, Valutazione: 4.5} | “Data non valida!” |  |
| 3 | Alunno intero lunghezza != 6 | Data valida, Alunno intero lunghezza != 6 [ERROR], Valutazione valida. |  | {Classe: “III A”, Data: “20/10/2022”, Alunno: 01, Valutazione: 4.5} | “Alunno non valido” |  |
| 4 | Valutazione valore < 1 | Data valida, Alunno valido, Valutazione float valore <1 [ERROR]. |  | {Classe: “III A”, Data: “20/10/2022”, Alunno: 000001, Valutazione: 0} | “Valutazione minore di 1” |  |
| 5 | Valutazione valore > 10 | Data valida, Alunno valido, Valutazione float valore >10 [ERROR] . |  | {Classe: “III A”, Data: “20/10/2022”, Alunno: 000001, Valutazione: 10.5} | “Valutazione maggiore di 10” |  |

**PIANO DI TEST UTILIZZANDO IL METODO DEL *CATEGORY-PARTITION TESTING* PER LA FUNZIONALITÀ “*ArrotondaVoto*”.**

|  |  |
| --- | --- |
| **ALUNNO** | **VALUTAZIONE** |
| * Intero di lunghezza = 6 * Intero di lunghezza != 6 [ERROR] | * Float con valori >=1 e <=10 * Float con valori < 1 [ERROR] * Float con valori >10 [ERROR] |

Il numero di test da effettuarsi senza particolari vincoli è:

Introduciamo i vincoli [ERROR].

Il numero di test da eseguire per testare singolarmente i vincoli è 3 (1 per Alunno, 2 per Valutazione).

Il numero di test risultante è: (1\*1) + 3 = 4

**TEST SUITE**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Descrizione** | **Classi di equivalenza coperte** | **Pre-condizioni** | **Input** | **Output Attesi** | **Post-condizioni Attese** |
| 1 | Tutti input validi | Data valida, Alunno valido, Valutazione valida. | È stata selezionata una classe esistente, l’alunno esiste e fa parte della classe selezionata, la pagella si trova nello stato di “non approvata”. | { Alunno: 000001, Valutazione: 7} | “Voto arrotondato” | La pagella viene aggiornata dal sistema attraverso *Aggiorna pagella* |
| 2 | Alunno intero lunghezza != 6 | Alunno intero lunghezza != 6 [ERROR], Valutazione valida. |  | { Alunno: 01, Valutazione: 7} | “Alunno non valido” |  |
| 3 | Valutazione valore < 1 | Alunno valido, Valutazione float valore <1 [ERROR]. |  | { Alunno: 000001, Valutazione: 0} | “Valutazione minore di 1” |  |
| 4 | Valutazione valore > 10 | Alunno valido, Valutazione float valore >10 [ERROR]. |  | { Alunno: 000001, Valutazione: 10.5} | “Valutazione maggiore di 10” |  |

**PIANO DI TEST UTILIZZANDO IL METODO DEL *CATEGORY-PARTITION TESTING* PER LA FUNZIONALITÀ “*InserisciMateria*”.**

|  |  |
| --- | --- |
| **NOME MATERIA** | **DOCENTE** |
| * Stringa di caratteri di lunghezza > 1 e <= 30 * Stringa di caratteri di lunghezza <= 1 [ERROR] * Stringa di caratteri di lunghezza >30 [ERROR] * Stringa che contiene simboli che non sono caratteri [ERROR] | * Stringa alfanumerica di lunghezza >0 e <=20 * Stringa vuota [ERROR] * Stringa di lunghezza > 20 [ERROR] |

Il numero di test da effettuarsi senza particolari vincoli è:

Introduciamo i vincoli [ERROR] .

Il numero di test da eseguire per testare singolarmente i vincoli è 5 (3 per NomeMateria, 2 per Docente).

Il numero di test risultante è: (1\*1) + 5 = 6

**TEST SUITE**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Descrizione** | **Classi di equivalenza coperte** | **Pre-condizioni** | **Input** | **Output Attesi** | **Post-condizioni Attese** |
| 1 | Tutti input validi | NomeMateria valido, NomeDocente valido | È stata selezionata una classe esistente, il docente esiste e insegna la materia inserita. | { NomeMateria: “Fisica”, Docente: “c.viola13”} | “Materia Aggiunta” | Alla classe viene aggiunta la materia |
| 2 | NomeMateria stringa <=1 | NomeMateria stringa <= 1 caratteri [ERROR], NomeDocente valido |  | { NomeMateria: “F”, Docente: “c.viola13”} | “Nome materia non valido!” |  |
| 3 | NomeMateria stringa > 30 | NomeMateria stringa >30 caratteri [ERROR], NomeDocente valido |  | { NomeMateria: “FisicaFisicaFisicaFisicaFisicaFisica”, Docente: “c.viola13”} | “Nome materia troppo lungo!” |  |
| 4 | NomeMateria stringa con simboli | NomeMateria stringa con simboli [ERROR], NomeDocente valido |  | { NomeMateria: “Fi$ic@”, Docente: “c.viola13”} | “Nome materia non valido!” |  |
| 5 | Docente stringa vuota | NomeMateria valido, NomeDocente stringa vuota [ERROR] |  | { NomeMateria: “Fisica”, Docente: “ ”} | “Username Docente vuoto!” |  |
| 6 | Docente stringa > 20 | NomeMateria valido, NomeDocente stringa > 20 caratteri [ERROR] |  | { NomeMateria: “Fisica”, Docente: “carla.viola.20.vvv.54 ”} | “Username Docente troppo lungo” |  |

# Progettazione

## Diagramma delle classi

Il progetto si struttura secondo un’architettura a livelli, cioè tutte le componenti che forniscono o danno l’accesso a un insieme di servizi correlati sono tenute insieme in un livello mentre tutto il resto ne è fuori. Quindi il livello boundary potrà accedere al livello immediatamente più basso, ma un livello basso non può accedere ai livelli più alti.

Immagine che contiene testo, schermata, diagramma, Parallelo

Descrizione generata automaticamenteDi seguito si riporta il diagramma delle classi di progettazione. Le classi sono raggruppate in package rispettando l’approccio BCED.

Il package Boundary è responsabile della visualizzazione dei dati sulla GUI. Il package Control disaccoppia la logica di presentazione dalla logica di business dell’applicazione. Il package Entity racchiude la logica di business e delega al package Database la realizzazione di tecniche di accesso ai dati persistenti, per cui ogni classe del livello Entity delega alla propria DAO (Data Access Object) la responsabilità di estrarre dati dal database.

Per reificare le classi associative Valutazione e GiudizioFinale, queste sono state trasformate nelle corrispondenti classi reali. Si rimanda al diagramma raffinato per un commento più approfondito sull’implementazione del software attraverso l’approccio BCED e l’utilizzo degli opportuni Design Patterns.

Immagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamenteNel BCED l’accesso al database viene gestito tramite la realizzazione di DAO, che richiamano tutti i metodi del DBManager di cui necessitano per effettuare le rispettive operazioni CRUD. Il DBManager gestisce quindi la connessione con il database che sono state realizzate con l’ausilio di MySQL Workbench.

Nella realizzazione del database si affronta il problema di tradurre le classi del Class Diagram in strutture non object-oriented e per farlo bisogna scegliere quali regole di traduzione applicare caso per caso.

Nel Database le scelte di progetto riguardano quindi la traduzione delle associazioni del Class Diagram in relazioni fra tabelle del database. Sono state fatte le seguenti traduzioni:

* **Associazione Uno-a-Uno fra genitori e studenti:**

la chiave primaria della tabella studenti è stata assegnata alla tabella genitori come chiave esterna.

* **Associazione Molti-a-Molti fra studenti e classi:**

la relazione è stata scomposta in due relazioni Uno-a-Molti utilizzando la tabella associativa *studenti\_in\_classe*.

* **Associazione uno a molti fra studenti e valutazioni:**

La tabella valutazioni ha come chiave esterna la chiave primaria della tabella studenti.

* **Associazione uno a molti fra studenti e pagelle:**

La tabella pagelle ha come chiave esterna la chiave primaria della tabella studenti.

* **Associazione uno a molti fra pagelle e giudizi finali:**

La tabella giudizi finali ha come chiave esterna la chiave primaria della tabella pagelle.

* **Associazione uno a molti fra docenti e materie:**

La tabella materie ha come chiave esterna la chiave primaria della tabella docenti.

* **Associazione uno a molti fra docente e attività:**

La tabella attività ha come chiave esterna la chiave primaria della tabella docenti.

* **Associazione uno a molti fra materie e valutazioni:**

La tabella valutazioni ha come chiave esterna la chiave primaria della tabella materie.

* **Associazione uno a molti fra materie e giudizi finali:**

La tabella giudizi finali ha come chiave esterna la chiave primaria della tabella materie.

* **Associazione uno a molti fra classi e materie:**

La tabella materie ha come chiave esterna la chiave primaria della tabella classi.

* **Associazione uno a uno fra classi e registro:**

La chiave primaria della tabella classi è stata assegnata alla tabella registri come chiave esterna.

* **Associazione uno a molti fra registro e attività:**

La tabella attività ha come chiave esterna la chiave primaria della tabella registro.

**Nota: Aumentare il numero di connection**

Avendo più viste collegate alle tabelle del database, l’applicazione necessitava di un buon numero di connections, è stato quindi modificato il numero massimo di connections con il Database dalla Shell di MySQL, così da non intaccare la visualizzazione nel boundary di liste estratte dal Database, che aumentano la user-friendliness del software.

La procedura è riportata di seguito:

Immagine che contiene testo, schermata, software, Carattere

Descrizione generata automaticamente

## Diagrammi di sequenza

Di seguito sono riportati i diagrammi di sequenza per due dei casi d’uso sviluppati.

* *AggiungiVoto*

Immagine che contiene testo, diagramma, Parallelo, schermata

Descrizione generata automaticamente

* *VisualizzaVoti*

Immagine che contiene testo, diagramma, Parallelo, linea

Descrizione generata automaticamente

# Implementazione

Immagine che contiene testo, schermata, Rettangolo, diagramma

Descrizione generata automaticamente

**Struttura del codice**

Per l’implementazione del codice è stata usata un’architettura a livelli seguendo il pattern BCED. Sono quindi presenti:

* Package Boundary: contiene una prima interfaccia MainFrame dalla quale è possibile aprire le schermate per il Docente, per lo Studente, per la Segreteria e per il Genitore. Da ciascuna di esse è possibile aprire ulteriori dialog per le funzionalità implementate. Ad esempio, da DialogDocente è possibile cliccare il bottone “Aggiungi Voto”, che porta alla finestra DialogAggiungiVoto, dalla quale è possibile utilizzare la funzionalità di inserimento di una valutazione ad uno studente.
* Package Control: contiene le azioni degli Use Case implementati. Ad esempio, la funzionalità aggiungiVoto.
* Package Entity: contiene le classi che rappresentano gli oggetti del dominio del sistema e memorizza i dati ottenuti dal package Database.
* Package Database: contiene gli oggetti responsabili dell’estrazione dei dati dal Database. È necessaria una classe DBManager, che fornisce i metodi di connessione e accesso/gestione al database.

Avendo la necessità di disaccoppiare il controller dalla logica di business del Package Entity, è stata realizzata una classe Façade EntityIstituto, che maschera al Controller i metodi di accesso alle Entity.  
Essendo questa classe Information Expert dell’applicazione, è stata realizzata anche la relativa classe DAO, in modo da accedere dal livello DAO al database per effettuare query per prelevare collezioni di dati non estraibili dalle DAO dei singoli oggetti (ad esempio: la lista delle classi, di studenti, di materie, ecc..) ed effettuare controlli su tali collezioni (ad esempio: l’esistenza di uno studente in una determinata classe, esistenza di un docente, verificare se una materia è insegnata da un determinato docente, ecc..).   
Per EntityIstituto e IstitutoDAO è stato utilizzato il design pattern Singleton, in modo da averne una sola istanza.

**Package explorer**

Di seguito è riportato il Package Explorer del progetto.

Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente Immagine che contiene testo, schermata, menu

Descrizione generata automaticamente

Il progetto è diviso in package che raggruppano tutti file appartenenti ai diversi livelli del modello BCED. Si è deciso inoltre di raggruppare in un package a parte tutte le eccezioni e le risorse utilizzate.

**Risorse del progetto**

Gli artefatti necessari per l’esecuzione del programma sono:

File Jar:   
– jcalendar.1-4.:  
– hamcrest-core\_1.3 Maven Central Repository   
– JUnit 4.13.2 Maven Central Repository  
Estensione di Eclipse per visualizzare il Control Flow Graph: <https://eclipsefcg.sourceforge.net/>

**Confronto stima dei costi**

Di seguito sono riportati il numero di LLOC scritte in Java:

* In totale si hanno 4993 LLOC

Di cui:

* 2629 per la funzionalità RegistraUtente
* 830 per la funzionalità AggiungiVoto
* 360 per la funzionalità Visualizza voti

Si noti che le LLOC definitive sono minori di quelle stimate nella stima dei costi.

I motivi della sovrastima iniziale sono:

* Il riutilizzo di librerie esterne per la creazione delle GUI.
* Un miglioramento dell’architettura che ha permesso di ridurre la complessità generale del progetto.
* Ottimizzazione del codice per quanto riguarda la riduzione di codice duplicato o inefficiente
* Eliminazione di funzionalità non necessarie.

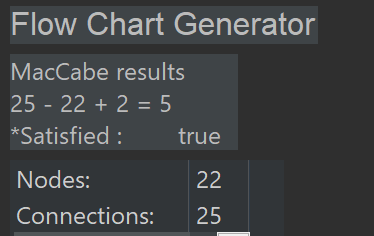
# Testing

## Test strutturale

## Complessità ciclomatica

Di seguito è riportato il codice Java del metodo aggiungiVoto della classe EntityIstituto, affiancato dal corrispondente Control Flow Graph, generato con l’estensione di Eclipse Control Flow Graph Generator.

|  |  |
| --- | --- |
| public int aggiungiVoto(String docente, int matricola, int idmaterie, Date data, float voto) {    //gli id vengono definiti dal sistema in modo che siano univoci  IstitutoDAO singleton = IstitutoDAO.*getInstance*();  int idvalutazione=singleton.getultimoidvalutazioni();  System.***out***.println(idvalutazione);  idvalutazione++; //variabile statica per assegnare univocamente l'id  System.***out***.println(idvalutazione);    EntityValutazione valutazione = new EntityValutazione();  valutazione.setIdvalutazioni(idvalutazione);      if(!singleton.esisteDocente(docente)) {  System.***out***.println("Docente non trovato");  return -1;  }    // Controllo se esiste uno studente con quella matricola  if (!singleton.esisteStudenteInClasse(matricola,idmaterie)) {  System.***out***.println("Studente non trovato");  return -1;  }  EntityStudente studente\_valutato = new EntityStudente(matricola);  valutazione.setStudente(studente\_valutato);  // Controllo se esiste una materia con idmaterie  if (!singleton.esisteMateriaInsegnata(idmaterie,docente)) {  System.***out***.println("Materia non trovata");  return -1;  }  EntityMateria materia\_valutata = new EntityMateria(idmaterie);  valutazione.setMateria(materia\_valutata);    // Controllo se la data è nel quadrimestre corrente  if (!singleton.isDataValida(data)) {  System.***out***.println("Data non valida");  return -1;  }      valutazione.setData(data);    try {  valutazione.setVoto(voto);  }catch(VotoNotValid e) {    return -2;  }      int ret = valutazione.scriviSuDB(idvalutazione);    return ret;  } |  |

Il calcolo del numero ciclomatico di McCabe V(G), con la formula V(G)=E – N + 2, dove E è il numero di archi in G e N è il numero di nodi in G, è il seguente.

Si ottiene lo stesso risultato con la formula V(G)=P + 1, dove P, numero dei nodi predicato, è 4, e con V(G)=RC+ 1, dove RC, il numero di regioni chiuse in G, è 4.

Accorpando in un unico nodo i nodi in sequenza, si ottiene il seguente Control Flow Graph:

Immagine che contiene catena

Descrizione generata automaticamente

I nodi predicato sono: 1, 3, 6, 9

I cammini indipendenti sono:

* C1: 0, 1, 3, 5, 6, 8, 9, 11
* C2: 0, 1, 2, 3, 5, 6, 8, 9, 11
* C3: 0, 1, 2, 3, 4, 5, 6, 8, 9, 11
* C4: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 11
* C5: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11

## Test funzionale

Per il test funzionale, si è deciso di utilizzare JUnit come framework di test unitari ed EclEmma come strumento di copertura del codice per testare gli input delle funzionalità “Registra Utente” e “Aggiungi Voto”, rifacendosi dunque ai test precedentemente pianificati nel Category-Partition Testing.

A causa delle varie dipendenze tra le diverse tabelle e i diversi controlli implementati, si è deciso di effettuare un test con “Classi” e “Materie” (senza docenti associati) come uniche tabelle popolate, per evitare conflitti con dati eventualmente inseriti prima.

Di seguito è riportato in verde il risultato positivo del test di coverage di EntityIstitutoTest.

package entity;

import static org.junit.Assert.\*;  
import org.junit.After;  
import org.junit.Before;  
import org.junit.Test;  
import java.util.ArrayList;  
import java.sql.Date;  
import java.time.LocalDate;  
import org.junit.FixMethodOrder;  
import org.junit.runners.MethodSorters;

*@FixMethodOrder*(*MethodSorters*.***NAME\_ASCENDING***)

public class EntityIstitutoTest {

*@Before*

public void setUp() throws Exception {

System.***out***.println("SETUP eseguito ad ogni caso di test");

}

*@After*

public void tearDown() throws Exception {

System.***out***.println("TEARDOWN eseguito ad ogni caso di test");

}

*@Test*

public void testA\_CreaDocente() {

EntityIstituto singleton = EntityIstituto.*getInstance*();

ArrayList<Integer> materie= new ArrayList<Integer>();

materie.add(1);

LocalDate localDate = LocalDate.*of*(1975,11,30);

java.sql.Date datanascita = java.sql.Date.*valueOf*(localDate);

int control=singleton.creaDocente("Luigi", "Bianchi", datanascita , "BNCLGU75S30H703D", "Salerno", "luigibianchi75@gmail.com", "+338763435", "l.bianchi30","BianchiPass3@" , materie);

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

int expected=1;

*assertEquals*(expected, control);

}

*@Test*

public void testB\_CreaStudente() {

EntityIstituto singleton = EntityIstituto.*getInstance*();

LocalDate localDate = LocalDate.*of*(2001,10,20);

java.sql.Date datanascita = java.sql.Date.*valueOf*(localDate);

int control=singleton.creaStudente("Mario", "Rossi", datanascita , "RSSMRA01R20F839P", "Napoli", "mariorossi2001@gmail.com", "338765432", "m.rossi20","ScuolaMario20@" , 1);

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

int expected=1;

*assertEquals*(expected, control);

}

*@Test*

public void testC\_CreaGenitore() {

EntityIstituto singleton = EntityIstituto.*getInstance*();

Date datanascita = new Date(1980-06-13);

//Test Case 1: Tutti input validi

int control=singleton.creaGenitore("Carla", "Viola", datanascita, "CRLVLI80H53F839Z","Napoli" , "carlaviola80@gmail.com", "327673435", "c.viola13", "IstitutoCarla80", 1);

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

int expected=1;

*assertEquals*(expected, control);

//Test Case 2: Nome string he inizia con una minuscola

control=singleton.creaGenitore("carla", "Viola", datanascita, "CRLVLI80H53F839Z","Napoli" , "carlaviola80@gmail.com", "327673435", "c.viola13", "IstitutoCarla80", 1);

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

expected=-2;

*assertEquals*(expected, control);

//Test Case 3: Nome vuoto

control=singleton.creaGenitore("", "Viola", datanascita, "CRLVLI80H53F839Z","Napoli" , "carlaviola80@gmail.com", "327673435", "c.viola13", "IstitutoCarla80", 1);

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

expected=-2;

*assertEquals*(expected, control);

//Test Case 4: Nome stringa di lunghezza >100

control=singleton.creaGenitore("Carla Carla Carla Carla Carla Carla Carla Carla Carla Carla Carla Carla Carla Carla Carla Carla Carla Carla Carla Carla Carla", "Viola", datanascita, "CRLVLI80H53F839Z","Napoli" , "carlaviola80@gmail.com", "327673435", "c.viola13", "IstitutoCarla80", 1);

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

expected=-2;

*assertEquals*(expected, control);

//Test Case 5: Cognome stringa di caratteri che inizia con una minuscola

control=singleton.creaGenitore("Carla", "viola", datanascita, "CRLVLI80H53F839Z","Napoli" , "carlaviola80@gmail.com", "327673435", "c.viola13", "IstitutoCarla80", 1);

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

expected=-5;

*assertEquals*(expected, control);

//Test Case 6:Cognome stringa vuota

control=singleton.creaGenitore("Carla", "", datanascita, "CRLVLI80H53F839Z","Napoli" , "carlaviola80@gmail.com", "327673435", "c.viola13", "IstitutoCarla80", 1);

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

expected=-5;

*assertEquals*(expected, control);

//Test Case 7:Cognome stringa di lunghezza >100

control=singleton.creaGenitore("Carla", "Viola Viola Viola Viola Viola Viola Viola Viola Viola Viola Viola Viola Viola Viola Viola Viola Viola Viola Viola Viola Viola", datanascita, "CRLVLI80H53F839Z","Napoli" , "carlaviola80@gmail.com", "327673435", "c.viola13", "IstitutoCarla80", 1);

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

expected=-5;

*assertEquals*(expected, control);

//Test Case 8: Codice Fiscale stringa alfanumerica di lunghezza != 16

control=singleton.creaGenitore("Carla", "Viola", datanascita, "CRLVLI80H53F839Z910","Napoli" , "carlaviola80@gmail.com", "327673435", "c.viola13", "IstitutoCarla80", 1);

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

expected=-6;

*assertEquals*(expected, control);

//Test Case 9: Codice Fiscale composta da simboli non validi

control=singleton.creaGenitore("Carla", "Viola", datanascita, "CR%L&80/5&F8$9ZF","Napoli" , "carlaviola80@gmail.com", "327673435", "c.viola13", "IstitutoCarla80", 1);

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

expected=-6;

*assertEquals*(expected, control);

//Test Case 10: Comune di residenza stringa vuota

control=singleton.creaGenitore("Carla", "Viola", datanascita, "CRLVLI80H53F839Z","" , "carlaviola80@gmail.com", "327673435", "c.viola13", "IstitutoCarla80", 1);

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

expected=-7;

*assertEquals*(expected, control);

//Test Case 11: Comune di Residenza stringa di lunghezza > 50

control=singleton.creaGenitore("Carla", "Viola", datanascita, "CRLVLI80H53F839Z","Napoli Napoli Napoli Napoli Napoli Napoli Napoli Napoli Napoli Napoli" , "carlaviola80@gmail.com", "327673435", "c.viola13", "IstitutoCarla80", 1);

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

expected=-7;

*assertEquals*(expected, control);

//Test Case 12: Comune di Residenza stringa composta da simboli non validi

control=singleton.creaGenitore("Carla", "Viola", datanascita, "CRLVLI80H53F839Z","N@p0li" , "carlaviola80@gmail.com", "327673435", "c.viola13", "IstitutoCarla80", 1);

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

expected=-7;

*assertEquals*(expected, control);

//Test Case 13: Email stringa in cui non è presente il simbolo @

control=singleton.creaGenitore("Carla", "Viola", datanascita, "CRLVLI80H53F839Z","Napoli" , "carlaviola80.gmail.com", "327673435", "c.viola13", "IstitutoCarla80", 1);

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

expected=-8;

*assertEquals*(expected, control);

//Test Case 14: Numero Cellulare Stringa vuota

control=singleton.creaGenitore("Carla", "Viola", datanascita, "CRLVLI80H53F839Z","Napoli" , "carlaviola80@gmail.com", "", "c.viola13", "IstitutoCarla80", 1);

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

expected=-9;

*assertEquals*(expected, control);

//Test Case 15: Numero Cellulare stringa di lunghezza > 15

control=singleton.creaGenitore("Carla", "Viola", datanascita, "CRLVLI80H53F839Z","Napoli" , "carlaviola80@gmail.com", "+39 33920182038402810 ", "c.viola13", "IstitutoCarla80", 1);

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

expected=-9;

*assertEquals*(expected, control);

//Test Case 16: Username stringa vuota

control=singleton.creaGenitore("Carla", "Viola", datanascita, "CRLVLI80H53F839Z","Napoli" , "carlaviola80@gmail.com", "327673435", "", "IstitutoCarla80", 1);

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

expected=-3;

*assertEquals*(expected, control);

//Test Case 17: Username stringa di lunghezza > 20

control=singleton.creaGenitore("Carla", "Viola", datanascita, "CRLVLI80H53F839Z","Napoli" , "carlaviola80@gmail.com", "327673435", "carla.viola13278738193037", "IstitutoCarla80", 1);

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

expected=-3;

*assertEquals*(expected, control);

//Test Case 18: Password stringa vuota

control=singleton.creaGenitore("Carla", "Viola", datanascita, "CRLVLI80H53F839Z","Napoli" , "carlaviola80@gmail.com", "327673435", "c.viola13", "", 1);

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

expected=-4;

*assertEquals*(expected, control);

//Test Case 19:Password stringa di lunghezza > 50

control=singleton.creaGenitore("Carla", "Viola", datanascita, "CRLVLI80H53F839Z","Napoli" , "carlaviola80@gmail.com", "327673435", "c.viola13", "IstitutoCarlaIstitutoCarlaIstitutoCarlaIstitutoCarlaIstitutoCarla", 1);

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

expected=-4;

*assertEquals*(expected, control);

}

*@Test*

public void testD\_AggiungiVoto() {

EntityIstituto singleton = EntityIstituto.*getInstance*();

// Preparazione dei dati di test

LocalDate localDate = LocalDate.*of*(2023, 06, 10);

java.sql.Date datavalutazione = java.sql.Date.*valueOf*(localDate);

float votoValido = (float)7.5;

// Esecuzione del metodo con un voto valido

int control = singleton.aggiungiVoto("l.bianchi30", 1, 1,datavalutazione, votoValido);

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

int expected=1;

*assertEquals*(expected, control); // Verifica che il risultato sia conforme alle aspettative

float votoMaggiore = 11;

// Esecuzione del metodo con un voto valido

control = singleton.aggiungiVoto("l.bianchi30", 1, 1,datavalutazione, votoMaggiore);

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

expected=-2;

*assertEquals*(expected, control); // Verifica che il risultato sia conforme alle aspettative

float votoMinore = (float) 0.5;

// Esecuzione del metodo con un voto valido

control = singleton.aggiungiVoto("l.bianchi30", 1, 1,datavalutazione, votoMinore);

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

expected=-2;

*assertEquals*(expected, control); // Verifica che il risultato sia conforme alle aspettative

}

}

Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente

Tutti i test hanno dato come esito PASS.

Immagine che contiene Software multimediale, software, Software per la grafica, Modifica

Descrizione generata automaticamente

Tutte le classi di equivalenza riguardanti le due funzioni sono stati coperti.

È stato raggiunto solo il 26,2% di coverage del codice, poiché sono state testate solo le righe di codice riguardanti “Registra Utente” e “Aggiungi Voto”. Oltretutto per il Category Partition Test si è deciso di testare tutti gli input non validi solo per uno dei tre Ruoli che può assumere l’utente, nel nostro caso il “Genitore”, poiché comuni a tutti.

**REGISTRA UTENTE**

A differenza dei test funzionali precedentemente pianificati, non sono stati implementati casi di test riguardanti le classi di equivalenza “Data di Nascita” e “Ruolo” poiché gestiti in maniera opportuna nel Boundary.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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 e Ruolo = “Docente” | Nome valido, Cognome valido, DataNascita valida, CodiceFiscale, ComuneResidenza valido, Email valida, NumeroCellulare valido, Username valido, Password valida, Ruolo = “Docente” valido | Username non registrato, Data valida, Materia esistente | {Nome: “Luigi”, Cognome: “Bianchi”, DataNascita: “30/11/1975”, CF: “BNCLGU75S30H703D”,  ComuneResidenza: “Salerno”, Email: [luigibianchi75@gmail.com](mailto:luigibianchi75@gmail.com),  NumeroCellulare: “338763435”, Username: “l.bianchi30”, Password: “BianchiPass3@”, Materia: “1”} | ret =1 | Docente aggiunto al Sistema con I relativi privilegi e le relative materie insegnata. | ret=1 | Docente aggiunto al sistema con le relative materie insegnate | PASS |
| 2 | Tutti input validi e Ruolo = “Alunno” | Nome valido, Cognome valido, DataNascita valida, CodiceFiscale, ComuneResidenza valido, Email valida, NumeroCellulare valido, Username valido, Password valida, Ruolo = “Alunno” valido | Username non registrato e classe esistente | {Nome: “Mario”, Cognome: “Rossi”, DataNascita: “20/10/2001”, CF: “RSSMRA01R20F839P”,  ComuneResidenza: “Napoli”, Email: [mariorossi2001@gmail.com](mailto:mariorossi2001@gmail.com),  NumeroCellulare: “338765432”, Username: “m.rossi20”, Password: “ScuolaMario20@”, Ruolo: “Alunno”, Classe: “1”} | ret=1 | Alunno aggiunto al Sistema con I relativi privilegi a cui viene assegnata una matricola e viene specificata la classe di appartenenza. | ret=1 | Alunno aggiunto al sistema con la matricola assegnata e la classe di appartenenza | PASS |
| 3 | Tutti input validi e Ruolo = “Genitore” | Nome valido, Cognome valido, DataNascita valida, CodiceFiscale, ComuneResidenza valido, Email valida, NumeroCellulare valido, Username valido, Password valida, Ruolo = “Genitore” valido | Username non registrato e matricola del figlio esistente e non associata a nessun genitore | {Nome: “Carla”, Cognome: “Viola”, DataNascita: “13/06/1980”, CF: “CRLVLI80H53F839Z”,  ComuneResidenza: “Napoli”, Email: [carlaviola80@gmail.com](mailto:carlaviola80@gmail.com),  NumeroCellulare: “327673435”, Username: “c.viola13”, Password: “IstitutoCarla80”, Ruolo: “Genitore”, Matricola: 1} | ret=1 | Genitore aggiunto al Sistema con I relativi privilegi e viene specificata la matricola dello studente di cui è genitore. | ret=1 | Genitore aggiunto al sistema e associato alla matricola del figlio inserita. | PASS |
| 4 | Nome stringa di caratteri che inizia con una minuscola | Nome stringa che inizia con minuscola [ERROR}, Cognome valido, DataNascita valida, CodiceFiscale, ComuneResidenza valido, Email valida, NumeroCellulare valido, Username valido, Password valida, Ruolo = “Genitore” valido |  | {Nome: “carla”, Cognome: “Viola”, DataNascita: “13/06/1980”, CF: “CRLVLI80H53F839Z”,  ComuneResidenza: “Napoli”, Email: [carlaviola80@gmail.com](mailto:carlaviola80@gmail.com),  NumeroCellulare: “327673435”, Username: “c.viola13”, Password: “IstitutoCarla”, Ruolo: “Genitore”, Matricola: 1} | ret=-2 e "Il nome deve iniziare con una maiuscola" |  | ret=-2 e "Il nome deve iniziare con una maiuscola" |  | PASS |
| 5 | Nome stringa vuota | Nome stringa vuota [ERROR}, Cognome valido, DataNascita valida, CodiceFiscale, ComuneResidenza valido, Email valida, NumeroCellulare valido, Username valido, Password valida, Ruolo = “Genitore” valido |  | {Nome: “ ”, Cognome: “Viola”, DataNascita: “13/06/1980”, CF: “CRLVLI80H53F839Z”,  ComuneResidenza: “Napoli”, Email: [carlaviola80@gmail.com](mailto:carlaviola80@gmail.com),  NumeroCellulare: “327673435”, Username: “c.viola13”, Password: “IstitutoCarla”, Ruolo: “Genitore”, Matricola: 1} | ret=-2 e "Nome vuoto" |  | ret=-2 e "Nome vuoto" |  | PASS |
| 6 | Nome stringa di lunghezza >100 | Nome stringa > 100 [ERROR}, Cognome valido, DataNascita valida, CodiceFiscale, ComuneResidenza valido, Email valida, NumeroCellulare valido, Username valido, Password valida, Ruolo = “Genitore” valido |  | {Nome: “ Carla Carla Carla Carla Carla Carla Carla Carla Carla Carla Carla Carla  Carla Carla Carla Carla Carla Carla Carla Carla Carla”, Cognome: “Viola”, DataNascita: “13/06/1980”, CF: “CRLVLI80H53F839Z”,  ComuneResidenza: “Napoli”, Email: [carlaviola80@gmail.com](mailto:carlaviola80@gmail.com),  NumeroCellulare: “327673435”, Username: “c.viola13”, Password: “IstitutoCarla”, Ruolo: “Genitore”, Matricola: 1} | ret=-2 e "Nome non valido" |  | ret=-2 e "Nome non valido" |  | PASS |
| 7 | Cognome stringa di caratteri che inizia con una minuscola | Nome valido, Cognome stringa che inizia con minuscola [ERROR}, DataNascita valida, CodiceFiscale, ComuneResidenza valido, Email valida, NumeroCellulare valido, Username valido, Password valida, Ruolo = “Genitore” valido |  | {Nome: “Carla”, Cognome: “viola”, DataNascita: “13/06/1980”, CF: “CRLVLI80H53F839Z”,  ComuneResidenza: “Napoli”, Email: [carlaviola80@gmail.com](mailto:carlaviola80@gmail.com),  NumeroCellulare: “327673435”, Username: “c.viola13”, Password: “IstitutoCarla”, Ruolo: “Genitore”, Matricola: 1} | ret=-5 e "Il cognome deve iniziare con una maiuscola" |  | ret=-5 e "Il cognome deve iniziare con una maiuscola" |  | PASS |
| 8 | Cognome stringa vuota | Nome valido, Cognome stringa vuota [ERROR}, DataNascita valida, CodiceFiscale, ComuneResidenza valido, Email valida, NumeroCellulare valido, Username valido, Password valida, Ruolo = “Genitore” valido |  | {Nome: “Carla”, Cognome: “ ”, DataNascita: “13/06/1980”, CF: “CRLVLI80H53F839Z”,  ComuneResidenza: “Napoli”, Email: [carlaviola80@gmail.com](mailto:carlaviola80@gmail.com),  NumeroCellulare: “327673435”, Username: “c.viola13”, Password: “IstitutoCarla”, Ruolo: “Genitore”, Matricola: 1} | ret=-5 e "cognome vuoto" |  | ret=-5 e "cognome vuoto" |  | PASS |
| 9 | Cognome stringa di lunghezza >100 | Nome valido, Cognome stringa > 100 [ERROR}, DataNascita valida, CodiceFiscale, ComuneResidenza valido, Email valida, NumeroCellulare valido, Username valido, Password valida, Ruolo = “Genitore” valido |  | {Nome: “Carla”, Cognome: “Viola Viola Viola Viola Viola Viola Viola Viola Viola Viola Viola Viola Viola Viola Viola Viola Viola Viola Viola Viola Viola”, DataNascita: “13/06/1980”, CF: “CRLVLI80H53F839Z”,  ComuneResidenza: “Napoli”, Email: [carlaviola80@gmail.com](mailto:carlaviola80@gmail.com),  NumeroCellulare: “327673435”, Username: “c.viola13”, Password: “IstitutoCarla”, Ruolo: “Genitore”, Matricola: 1} | ret=-5 e "cognome non valido" |  | ret=-5 e "cognome non valido" |  | PASS |
| 10 | Codice Fiscale stringa alfanumerica di lunghezza != 16 | Nome valido, Cognome valido ,DataNascita valida, CodiceFiscale stringa != 16 [ERROR], ComuneResidenza valido, Email valida, NumeroCellulare valido, Username valido, Password valida, Ruolo = “Genitore” valido |  | {Nome: “Carla”, Cognome: “Viola”, DataNascita: “13/06/1980”, CF: “CRLVLI80H53F839Z910”,  ComuneResidenza: “Napoli”, Email: [carlaviola80@gmail.com](mailto:carlaviola80@gmail.com),  NumeroCellulare: “327673435”, Username: “c.viola13”, Password: “IstitutoCarla”, Ruolo: “Genitore”, Matricola: 1} | ret=-6 e "Codice fiscale: lunghezza non valida" |  | ret=-6 e "Codice fiscale: lunghezza non valida" |  | PASS |
| 11 | Codice Fiscale composta da simboli non validi | Nome valido, Cognome valido ,DataNascita valida, CodiceFiscale stringa != 16 [ERROR], ComuneResidenza valido, Email valida, NumeroCellulare valido, Username valido, Password valida, Ruolo = “Genitore” valido |  | {Nome: “Carla”, Cognome: “Viola”, DataNascita: “13/06/1980”, CF: “CR%L&80/5&F8$9ZF  ”,  ComuneResidenza: “Napoli”, Email: [carlaviola80@gmail.com](mailto:carlaviola80@gmail.com),  NumeroCellulare: “327673435”, Username: “c.viola13”, Password: “IstitutoCarla”, Ruolo: “Genitore”, Matricola: 1} | ret=-6 e "Codice fiscale: formato non valido" |  | ret=-6 e "Codice fiscale: formato non valido" |  | PASS |
| 12 | Comune di residenza stringa vuota | Nome valido, Cognome valido, DataNascita valida, CodiceFiscale, ComuneResidenza stringa vuota [ERROR], Email valida, NumeroCellulare valido, Username valido, Password valida, Ruolo = “Genitore” valido |  | {Nome: “Carla”, Cognome: “Viola”, DataNascita: “13/06/1980”, CF: “CRLVLI80H53F839Z”,  ComuneResidenza: “ ”, Email: [carlaviola80@gmail.com](mailto:carlaviola80@gmail.com),  NumeroCellulare: “327673435”, Username: “c.viola13”, Password: “IstitutoCarla”, Ruolo: “Genitore”, Matricola: 1} | ret=-7 e "Comune di residenza: vuoto" |  | ret=-7 e "Comune di residenza: vuoto" |  | PASS |
| 13 | Comune di Residenza stringa di lunghezza > 50 | Nome valido, Cognome valido, DataNascita valida, CodiceFiscale, ComuneResidenza stringa > 50 [ERROR], Email valida, NumeroCellulare valido, Username valido, Password valida, Ruolo = “Genitore” valido |  | {Nome: “Carla”, Cognome: “Viola”, DataNascita: “13/06/1980”, CF: “CRLVLI80H53F839Z”,  ComuneResidenza: “Napoli Napoli Napoli Napoli Napoli Napoli Napoli Napoli Napoli”, Email: [carlaviola80@gmail.com](mailto:carlaviola80@gmail.com),  NumeroCellulare: “327673435”, Username: “c.viola13”, Password: “IstitutoCarla”, Ruolo: “Genitore”, Matricola: 1} | ret=-7 e "Comune di residenza: lunghezza non valida" |  | ret=-7 e "Comune di residenza: lunghezza non valida" |  | PASS |
| 14 | Comune di Residenza stringa composta da simboli non validi | Nome valido, Cognome valido, DataNascita valida, CodiceFiscale, ComuneResidenza stringa con simboli non validi [ERROR], Email valida, NumeroCellulare valido, Username valido, Password valida, Ruolo = “Genitore” valido |  | {Nome: “Carla”, Cognome: “Viola”, DataNascita: “13/06/1980”, CF: “CRLVLI80H53F839Z”,  ComuneResidenza: “N@p0li”, Email: [carlaviola80@gmail.com](mailto:carlaviola80@gmail.com),  NumeroCellulare: “327673435”, Username: “c.viola13”, Password: “IstitutoCarla”, Ruolo: “Genitore”, Matricola: 1} | ret=-7 e "Comune di residenza: formato non valido" |  | ret=-7 e "Comune di residenza: formato non valido" |  | PASS |
| 15 | Email stringa in cui non è presente il simbolo @ | Nome valido, Cognome valido, DataNascita valida, CodiceFiscale, ComuneResidenza valido, Email stringa in cui non è presente @ [ERROR], NumeroCellulare valido, Username valido, Password valida, Ruolo = “Genitore” valido |  | {Nome: “Carla”, Cognome: “Viola”, DataNascita: “13/06/1980”, CF: “CRLVLI80H53F839Z”,  ComuneResidenza: “Napoli”, Email: [carlaviola80.gmail.com](mailto:carlaviola80@gmail.com),  NumeroCellulare: “327673435”, Username: “c.viola13”, Password: “IstitutoCarla”, Ruolo: “Genitore”, Matricola: 1} | ret=-8 e "Email non valida" |  | ret=-8 e " Email non valida" |  | PASS |
| 16 | Numero Cellulare Stringa vuota | Nome valido, Cognome valido, DataNascita valida, CodiceFiscale, ComuneResidenza valido, Email valida, NumeroCellulare stringa vuota [ERROR], Username valido, Password valida, Ruolo = “Genitore” valido |  | {Nome: “Carla”, Cognome: “Viola”, DataNascita: “13/06/1980”, CF: “CRLVLI80H53F839Z”,  ComuneResidenza: “Napoli”, Email: [carlaviola80@gmail.com](mailto:carlaviola80@gmail.com),  NumeroCellulare: “ ”, Username: “c.viola13”, Password: “IstitutoCarla”, Ruolo: “Genitore”, Matricola: 1} | ret=-9 e "Numero di cellulare vuoto" |  | ret=-9 e " Numero di cellulare vuoto" |  | PASS |
| 17 | Numero Cellulare stringa di lunghezza > 15 | Nome valido, Cognome valido, DataNascita valida, CodiceFiscale, ComuneResidenza valido, Email valida, NumeroCellulare stringa > 15 [ERROR], Username valido, Password valida, Ruolo = “Genitore” valido |  | {Nome: “Carla”, Cognome: “Viola”, DataNascita: “13/06/1980”, CF: “CRLVLI80H53F839Z”,  ComuneResidenza: “Napoli”, Email: [carlaviola80@gmail.com](mailto:carlaviola80@gmail.com),  NumeroCellulare: “+39 33920182038402810 ”, Username: “c.viola13”, Password: “IstitutoCarla”, Ruolo: “Genitore”, Matricola: 1} | ret=-9 e "Numero di cellulare non valido: numero troppo lungo" |  | ret=-9 e " Numero di cellulare non valido: numero troppo lungo" |  | PASS |
| 18 | Username stringa vuota | Nome valido, Cognome valido, DataNascita valida, CodiceFiscale, ComuneResidenza valido, Email valida, NumeroCellulare valido, Username stringa vuota [ERROR], Password valida, Ruolo = “Genitore” valido |  | {Nome: “Carla”, Cognome: “Viola”, DataNascita: “13/06/1980”, CF: “CRLVLI80H53F839Z”,  ComuneResidenza: “Napoli”, Email: [carlaviola80@gmail.com](mailto:carlaviola80@gmail.com),  NumeroCellulare: “327673435”, Username: “ ”, Password: “IstitutoCarla”, Ruolo: “Genitore”, Matricola: 1} | ret=-3 e "Username vuoto" |  | ret=-3 e "Username vuoto" |  | PASS |
| 19 | Username stringa di lunghezza > 20 | Nome valido, Cognome valido, DataNascita valida, CodiceFiscale, ComuneResidenza valido, Email valida, NumeroCellulare valido, Username stringa > 20 [ERROR], Password valida, Ruolo = “Genitore” valido |  | {Nome: “Carla”, Cognome: “Viola”, DataNascita: “13/06/1980”, CF: “CRLVLI80H53F839Z”,  ComuneResidenza: “Napoli”, Email: [carlaviola80@gmail.com](mailto:carlaviola80@gmail.com),  NumeroCellulare: “327673435”, Username: “carla.viola13278738193037”, Password: “IstitutoCarla”, Ruolo: “Genitore”, Matricola: 1} | ret=-3 e "Username troppo lungo" |  | ret=-3 e "Username troppo lungo" |  | PASS |
| 20 | Password stringa vuota | Nome valido, Cognome valido, DataNascita valida, CodiceFiscale, ComuneResidenza valido, Email valida, NumeroCellulare valido, Username valido, Password stringa vuota [ERROR], Ruolo = “Genitore” valido |  | {Nome: “Carla”, Cognome: “Viola”, DataNascita: “13/06/1980”, CF: “CRLVLI80H53F839Z”,  ComuneResidenza: “Napoli”, Email: [carlaviola80@gmail.com](mailto:carlaviola80@gmail.com),  NumeroCellulare: “327673435”, Username: “c.viola13”, Password: “ ”, Ruolo: “Genitore”, Matricola: 1} | ret=-4 e "Password vuoto" |  | ret=-4 e "Password vuoto" |  | PASS |
| 21 | Password stringa di lunghezza  > 50 | Nome valido, Cognome valido, DataNascita valida, CodiceFiscale, ComuneResidenza valido, Email valida, NumeroCellulare valido, Username valido, Password stringa > 50[ERROR], Ruolo = “Genitore” valido |  | {Nome: “Carla”, Cognome: “Viola”, DataNascita: “13/06/1980”, CF: “CRLVLI80H53F839Z”,  ComuneResidenza: “Napoli”, Email: [carlaviola80@gmail.com](mailto:carlaviola80@gmail.com),  NumeroCellulare: “327673435”, Username: “c.viola13”, Password: “IstitutoCarlaIstitutoCarlaIstitutoCarlaIstitutoCarlaIstitutoCarla”, Ruolo: “Genitore”, Matricola: 1} | ret=-4 e "Password troppo lunga" |  | ret=-4 e "Password troppo lunga" |  | PASS |

Le diverse precondizioni sono gestite evisualizzate nel Boundary. Se non rispettate non permettono la registrazione dell’utente**.**

**NOTE:** Per i diversi formati non validi si è deciso di utilizzare le Regex, che hanno permesso di specificare alcune regole di ricerca di caratteri speciali nelle stringhe

**AGGIUNGI VOTO**

Anche in questo caso la Data non è stata testata poiché l’input viene gestito nel Boundary. Inoltre, differentemente dal piano di test, la matricola non ha una lunghezza predefinita pari a 6 ed inoltre viene gestita anch’essa nel Boundary attraverso diversi controlli.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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 | Data valida, Alunno valido, Valutazione valida. | È stata selezionata una classe esistente; l’alunno esiste e fa parte della classe selezionata; il docente insegna una materia nella classe selezionata | { Data: “10/06/2023” , Docente: l.bianchi30, Alunno: 1, Materia 1, Valutazione: 7.5} | ret=1 | La valutazione viene inserita nel registro elettronico e a causa della valutazione negativa il sistema invia una email al genitore dell’alunno. | ret=1 | La valutazione viene aggiunta al database ed è visualizzabile dal genitore dell’alunno | PASS |
| 2 | Valutazione valore < 1 | Data valida, Alunno valido, Valutazione float valore <1 [ERROR]. |  | { Data: “10/06/2023” , Docente: l.bianchi30, Alunno: 1, Materia 1, Valutazione: 0.5} | ret=-2 e “Il voto deve essere almeno pari a 1!” |  | ret=-2 e “Il voto deve essere almeno pari a 1!” |  | PASS |
| 3 | Valutazione valore > 10 | Data valida, Alunno valido, Valutazione float valore >10 [ERROR] . |  | { Data: “10/06/2023” , Docente: l.bianchi30, Alunno: 1, Materia 1, Valutazione: 11} | ret=-2 e “Il voto deve essere al massimo pari a 10!” |  | ret=-2 e “Il voto deve essere al massimo pari a 10!” |  | PASS |

Le diverse precondizioni sono gestite e visualizzate nel Boundary. Se non rispettate non permettono l’inserimento del voto.