Task 4 – Prima Iterazione

Gruppo G4: Arena Letizia M63001513

Ferrara Leonardo M63001517

1. Descrizione del task:

Requisiti sul mantenimento delle Partite giocate:

Per ogni partita giocata dal giocatore il sistema deve mantenere lo storico di tale partita, memorizzando l'Id del giocatore, il tipo di partita (primo scenario, secondo scenario...) la data e l'ora di inizio e di termine della partita, la classe testata, l'insieme dei casi di test creati e i relativi risultati, nonché il Robot con cui si è giocato ed i casi di test creati dal Robot con i relativi risultati.

2. Analisi esplorativa dei requisiti:

Storie utente

Di seguito proviamo ad immaginare come i task interni all'applicazione possano voler interagire con il task da noi sviluppato. Si noti che in questo caso per utente non si intende lo studente, utente finale dell'applicazione, ma i team deputati allo sviluppo dei task che usufruiscono dei servizi offerti da questo task.

Procederemo applicando la seguente formula: "Come [utente], voglio che [descrizione del servizio], in modo da [valore restituito dal servizio]."

Le elenchiamo di seguito:

- Come sviluppatore del task T5, voglio poter salvare la partita creata e i relativi dati in modo da permettere ad altri task di tenerne traccia.

- Come sviluppatore del task T6, voglio poter allocare i test case scritti in un database in modo da permettere agli altri task di procedere con la loro compilazione ed esecuzione.

- Come sviluppatore del task T7, voglio poter allocare in un database il risultato della compilazione, ed eventualmente l'esito (in termini di fault coverage), dei test case elaborati in modo da permetterne il recupero successivamente.

- Come sviluppatore del task T8, voglio poter allocare i test case scritti dal robot in un database in modo da permettere agli altri task di procedere con la loro compilazione ed esecuzione.

- Come sviluppatore del task T9, voglio poter allocare i test case scritti dal robot in un database in modo da permettere agli altri task di procedere con la loro compilazione ed esecuzione.

Revisione dei requisiti:

1. Il sistema deve memorizzare lo storico di ogni partita giocata dal giocatore.
2. Ogni partita è contrassegnata da: Id del giocatore, tipo di partita (primo scenario, secondo scenario...), data e ora di inizio e di termine della partita, classe testata, insieme dei casi di test creati e relativi risultati, Robot con cui si è giocato e casi di test creati dal Robot con relativi risultati.

Classificazione dei requisiti:

Requisiti funzionali

ID	Requisito	Origine (n. frase dei requisiti revisionati)
RF01	Il sistema deve memorizzare lo storico di ogni partita giocata	1

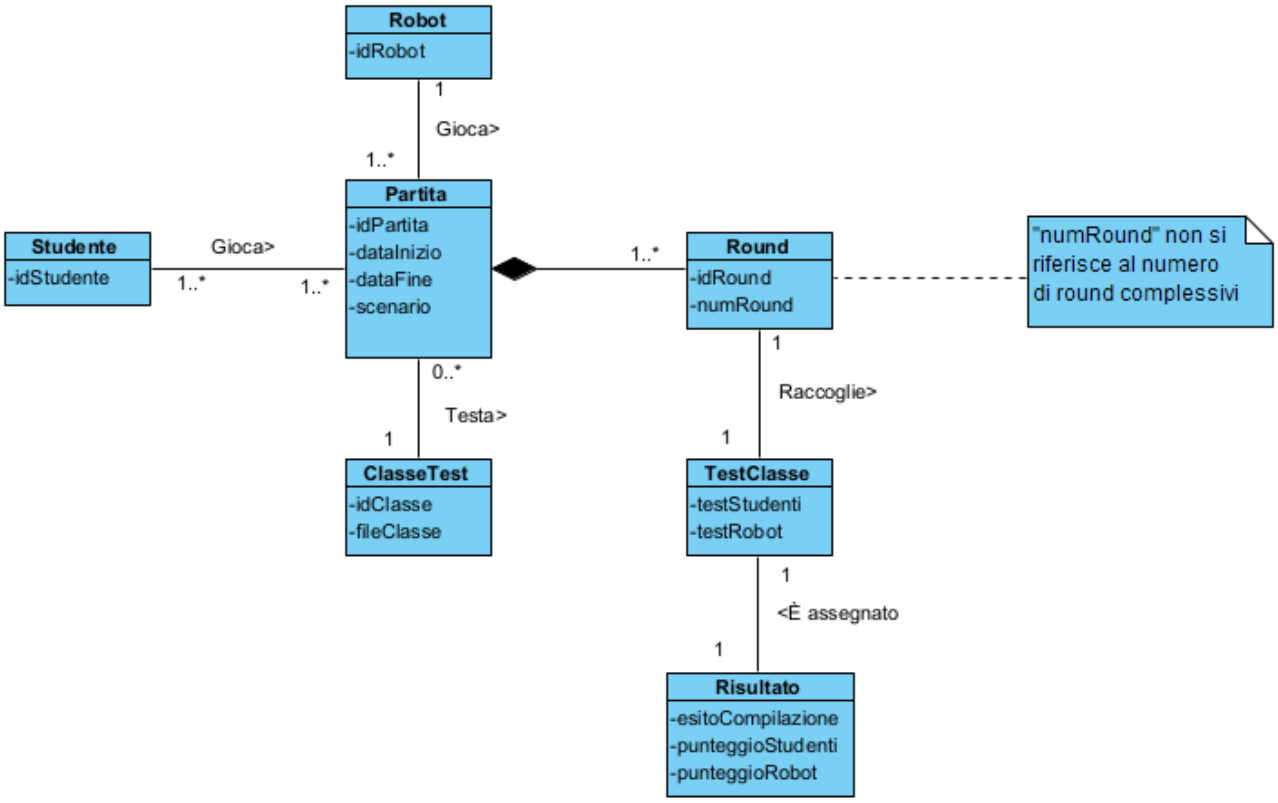
Requisiti sui dati

ID	Requisito	Origine (n. frase dei requisiti revisionati)
RD01	Ogni partita è contrassegnata da tipo di partita, data e ora di inizio e di termine della partita, classe testata, dati relativi al giocatore e dati relativi al robot.	2
RD02	I risultati sono espressi in termini di punteggi ottenuti dal giocatore e dal robot.	2
RD03	I dati relativi al giocatore sono il suo identificativo, l'insieme dei casi di test da esso creati, ed i risultati ottenuti.	2
RD04	I dati relativi al robot sono il suo identificativo, l'insieme dei casi di test da esso creati, ed i risultati ottenuti.	2

Vincoli

ID	Requisito	Origine (n. frase dei requisiti revisionati)
V01	Sono possibili solo tre tipi di partita: primo, secondo e terzo scenario	

Diagramma delle classi (di analisi)

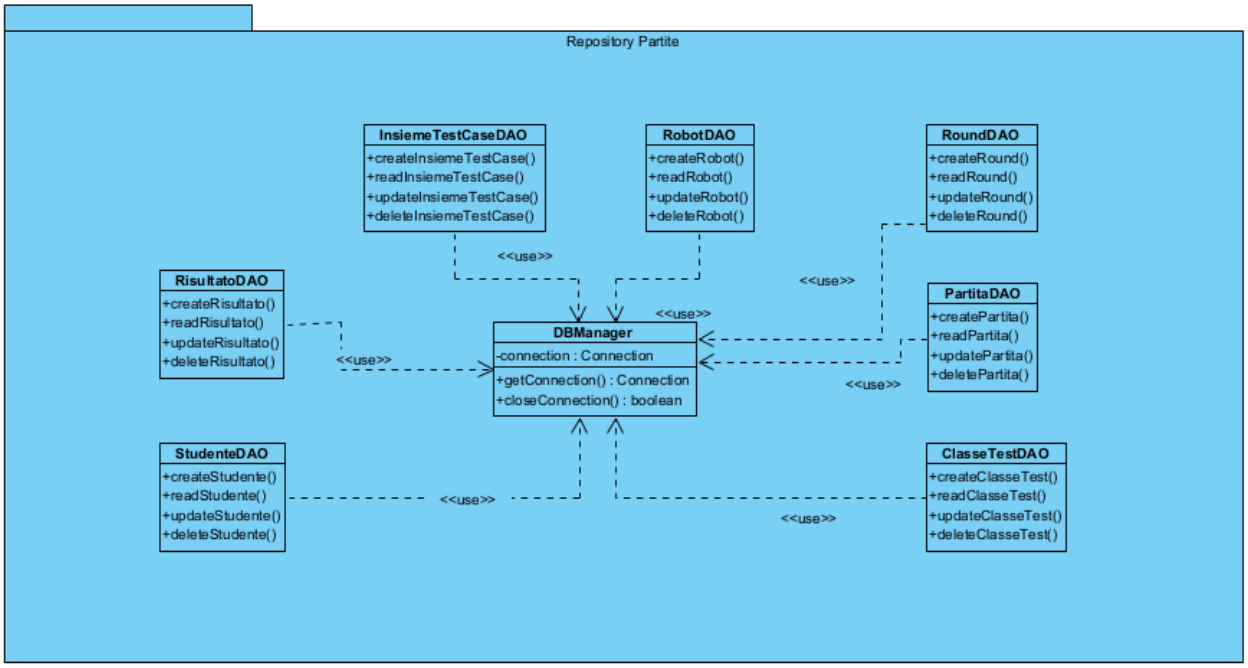


3. Modello concettuale dei dati & glossario:

Termine	Descrizione	Tipo
IdPartita	Identificativo della partita giocata	objectId
IdStudente	Identificativi dello studente che gioca la partita	array di string
IdRobot	Identificativo dello strumento automatico contro cui andrà a giocare lo studente, da lui scelto	string
dataInizio/ dataFine	Date di inizio e di conclusione della partita	date
scenario	Scenario di gioco che può essere di tre tipi: - Singolo Turno, Singolo Avversario, Singola Classe da testare - Multipli Turni (Numero prefissato), Singolo Avversario, Singola Classe da testare - Multipli Turni (Numero prefissato), Multipli giocatori, Singola Classe da testare	int
IdClasse	Identificativo della classe Java da testare selezionata dallo studente	string

fileClasse	Corpo della classe selezionata dallo studente per la partita	string
testStudente	Array che contiene il corpo dei test case scritti dagli studenti	array di string
testRobot	Corpo del test case scritto dal robot	string
idRound	Identificativo numerico del singolo round di una partita	objectId
numRound	Numero del round giocato all'interno della partita	int
esitoCompilazione	Esiti con criterio pass/fail della compilazione dell'insieme di casi di test sottomessi dagli studenti nel round	array di boolean
punteggioStudente	Punteggi in termini di fault coverage ottenuti eseguendo i casi di test sottomessi dagli studenti (compreso tra 0 e 100)	array di int
punteggioRobot	Punteggio in termini di fault coverage ottenuto eseguendo i casi di test sottomessi dal robot (compreso tra 0 e 100)	int

4. Diagramma delle classi (di progettazione):



Abbiamo realizzato un diagramma delle classi che mostri come gli oggetti del repository siano accessibili dalle apposite interfacce, che offrono funzionalità CRUD. Abbiamo pensato di inserire queste classi all'interno di un package che definisca un raggruppamento logico di elementi semanticamente collegati fra di loro.

5. Studio delle tecnologie necessarie:

Lo sviluppo dei servizi verrà effettuato in Java, sfruttando l'ambiente di sviluppo Eclipse, con il quale il team ha già familiarità grazie ad esperienze maturate precedentemente.

Per permettere la collaborazione tra i membri del team, oltre che il collegamento dei task sviluppati dagli altri team, verrà sfruttato un repository Git online, in modo che utenti multipli possano lavorare contemporaneamente.

Per il mantenimento dei dati, visto il loro formato variabile, abbiamo deciso di optare per un database di tipo documentale. Le basi di dati documentali sono utilizzate per la gestione di dati semi-strutturati (dati che non hanno una struttura definita, ma contengono etichette o marcatori per classificare le informazioni), come nel nostro caso.
Nei database documentali i dati sono memorizzati in documenti (oggetti JSON). Ciascun documento contiene coppie di campi e valori: i nomi dei campi permettono di capire a prima vista quali tipi di dato sono contenuti nel documento, mentre i valori possono essere di vari tipi (stringhe, array, oggetti, etc) e le loro strutture si allineano con gli oggetti usati dagli sviluppatori nel codice, semplificando quindi l'attività di programmazione. Durante la ricerca delle informazioni vengono esaminati i documenti stessi: le varie colonne con i dati di interesse non vengono cercate nel database, bensì i dati vengono estratti direttamente dal documento.
I vantaggi ottenuti da questa scelta, rispetto a quella di utilizzare un database relazionale, sono dunque la possibilità di evitare la definizione a monte delle tabelle del database, la possibilità di effettuare una mappatura 1:1 dei dati con gli oggetti usati nel codice, e l'alta flessibilità del sistema.

In particolare, abbiamo scelto di utilizzare la base di dati open-source MongoDB, il database documentale più utilizzato e popolare sul mercato.
Questo ha un modello di dati flessibile che permette di memorizzare dati non strutturati, offre pieno supporto per l'indicizzazione, e permette infine l'uso di API (Application Programming Interface) complete ed intuitive.
Infine, delle possibili opzioni proposte abbiamo scelto la versione cloud del software, MongoDB Atlas, in modo da decentralizzare il database, non restringendolo ad una macchina fisica, così da facilitare i processi di integrazione con gli altri task.

6. Prototipi:

Allo scopo di meglio comprendere i requisiti, abbiamo creato dei prototipi di documento sulla base di dati scelta. In particolare, abbiamo deciso di sviluppare prototipi per due situazioni diverse: quella di una partita del primo scenario, ossia singolo giocatore che gioca un singolo round, e quella di una partita del terzo scenario, con tre giocatori che giocano due round.

Nonostante nei database documentali sia solito conservare tutti i dati in una singola collection di documenti, abbiamo scelto di creare due collection: una che contenesse i documenti relativi alle partite e l'altra che contenesse i documenti relativi ai singoli round, che mantengono il riferimento alla partita di appartenenza mediante l'attributo idPartita. Questa scelta è stata fatta sia per motivi di chiarezza che per conformarsi all'architettura C&C mostrata nella presentazione del caso di studio, che indica come interfacce CRUDGame e CRUDRound.

Documento relativo alla partita con il 1° scenario:


```
_id: ObjectId('643c2650dfa2563ce09d8a47')
idStudente: Array
  0: "M63001517"
idRobot: "Randoop"
dataInizio: 2023-04-16T16:00:00.000+00:00
idClasse: "Persona"
testoClasse: "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do
            eiusm…"
dataFine: 2023-04-16T16:55:00.000+00:00
scenario: 1
```


Documento relativo al suo unico round:


```
_id: ObjectId('643d3108dfa2563ce0255489')
numRound: 1
testStudente: Array
  0: "Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusan…"
testRobot: "Neque porro quisquam est, qui dolorem ipsum quia dolor sit amet, conse…"
esitoCompilazione: true
punteggioRobot: 97
punteggioStudenti: Array
  0: 85
idPartita: ObjectId('643c2650dfa2563ce09d8a47')
```


Documento relativo alla partita con il 3° scenario:


```
_id: ObjectId('643d330f4cd4642fa9e83319')
idStudente: Array
  0: "M63001517"
  1: "M63001513"
  2: "M63001980"
idRobot: "Evosuite"
dataInizio: 2023-04-17T13:54:00.000+00:00
idClasse: "Persona"
testoClasse: "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do
            eiusm…"
dataFine: 2023-04-16T16:26:00.000+00:00
scenario: 3
```

Documento relativo al primo round della partita con il 3° scenario:


```
_id: ObjectId('643d33b14cd4642fa9e8331a')
numRound: 1
▾ testStudente: Array
    0: "Sed ut perspiciatis unde omnis iste natus error sit voluptatem
       accusa…"
    1: "Quis autem vel eum iure reprehenderit qui in ea voluptate velit
       esse …"
    2: "But I must explain to you how all this mistaken idea of
       denouncing pl…"
  testRobot: "Neque porro quisquam est, qui dolorem ipsum quia dolor sit amet,
              cons…"
▾ esitoCompilazione: Array
    0: true
    1: true
    2: false
  punteggioRobot: 93
▾ punteggioStudenti: Array
    0: 80
    1: 77
    2: 0
  idPartita: ObjectId('643d330f4cd4642fa9e83319')
```


Documento relativo al secondo round della partita con il 3° scenario:


```
_id: ObjectId('643d34ab4cd4642fa9e8331b')
numRound: 2
▾ testStudente: Array
    0: "At vero eos et accusamus et iusto odio dignissimos ducimus qui blandit…"
    1: "Et harum quidem rerum facilis est et expedita distinctio. Nam libero t…"
    2: "Temporibus autem quibusdam et aut officiis debitis aut rerum necessita…"
  testRobot: "Itaque earum rerum hic tenetur a sapiente delectus, ut aut reiciendis …"
▾ esitoCompilazione: Array
    0: true
    1: true
    2: true
  punteggioRobot: 65
▾ punteggioStudenti: Array
    0: 50
    1: 67
    2: 88
  idPartita: ObjectId('643d330f4cd4642fa9e83319')
```


7. Diario del team:

Indichiamo di seguito l'effort dedicato ad ognuna delle attività mostrate:

- Analisi esplorativa dei requisiti: 4 ore
- Modello concettuale dei dati e glossario: 2 ore
- Diagramma delle classi di progettazione: 30 minuti
- Studio delle tecnologie necessarie: 2 ore e 30 minuti
- Prototipi: 1 ora