



Documentazione Quiz Game

1 Indice

1	Indice	2
2	Introduzione	4
2.1	Informazioni sul progetto	4
2.2	Scopo	4
3	Analisi	5
3.1	Analisi del dominio	5
3.2	Analisi e specifica dei requisiti	5
3.3	Use case	8
3.4	Pianificazione	9
3.5	Analisi dei mezzi	10
3.5.1	Software	10
3.5.2	Librerie	10
3.5.3	Hardware	10
4	Progettazione	11
4.1	Design dell'architettura del sistema	11
4.2	Design delle interfacce	12
4.3	Design procedurale	16
4.3.1	Diagramma di flusso	16
4.3.2	Diagramma delle classi	17
5	Implementazione	18
5.1	Struttura cartelle	18
5.2	Classi Principali	18
5.2.1	Giocatore	18
5.2.2	Domanda	19
5.2.3	QuizGame	19
5.3	Interfaccia Grafica	20
5.3.1	Principale	20
5.3.2	Menu Iniziale	20
5.4	Impostazioni	21
5.5	Domande	21
5.5.1	Vero/Falso	21
5.5.2	Scelta Multipla	22
5.6	Classifica	22
5.7	quiz.json	23
5.7.1	Contenitore vero/falso	23
5.7.2	Contenitore scelta Multipla	23
6	Protocollo di test	24
6.1	Risultati test	27
6.2	Mancanze/limitazioni conosciute	27
7	Consuntivo	28
8	Conclusioni	29
8.1	Sviluppi futuri	29
8.2	Considerazioni personali	29
9	Bibliografia	30
9.1	Sitografia	30
10	Glossario	31

	SAMT – Sezione Informatica	Pagina 3 di 31
	Documentazione Quiz Game	

11	Indice delle figure.....	31
12	Allegati.....	31

	SAMT – Sezione Informatica	Pagina 4 di 31
	Documentazione Quiz Game	

2 Introduzione

2.1 Informazioni sul progetto

Allievi: Andrea Casamatta

Docente responsabile: Geo Petrini


SAMT Sezione Informatica 3BB modulo 306

Inizio: 04.09.2024

Termine: 01.01.2024

2.2 Scopo

Questo progetto è un'applicazione Java. L'applicativo ha lo scopo di far imparare e divertire la gente, essendo che si può giocare contro altre persone localmente, può essere anche usato per imparare assieme un po' di cultura generale. L'applicativo ha diverse categorie di domande e diverse difficoltà per essere più vario, le risposte potranno avere diverse modalità. Ogni utente avrà il proprio nome e ci sarà una classifica dei giocatori con i punteggi di ognuno di essi.

	SAMT – Sezione Informatica	Pagina 5 di 31
	Documentazione Quiz Game	

3 Analisi

3.1 Analisi del dominio

Questo prodotto serve per far domande generali che hanno lo scopo di far imparare e divertire allo stesso tempo i giocatori di tutte le età. Simile al gioco Trivia Quiz, solo che si gioca in locale e non online, con un minimo di 1 giocatore e il massimo di 10 giocatori. L'applicazione è sviluppata in Java, nell'editor di Netbeans. Per creare questo applicativo è necessario sapere usare l'interfaccia di Java in Netbeans e avere delle conoscenze su come programmare in Java.

3.2 Analisi e specifica dei requisiti

Il quiz game deve soddisfare diversi bisogni del committente, tra cui divertimento e apprendimento. Le funzioni principali del prodotto includono categorie di domande come geografia, storia, lingue, materie scolastiche e sport, con livelli di difficoltà (facile, medio, difficile) che assegnano punti diversi e include anche la possibilità di giocare in più persone. Il sistema di ranking deve mostrare la classifica finale con punteggio e nome utente. Le risposte possono essere vero/falso o a scelta multipla, e le domande possono includere solo testi.

Per implementare queste funzioni, sono necessari file per memorizzare domande e risposte.

Requisito	Req-01	Priorità	1	Versione	1.0
Nome	Categorie domande				
Note	L'applicativo deve supportare la creazione e gestione di 5 categorie di domande: Geografia, Storia, Lingue, Materie scolastiche e Sport.				
Sotto requisiti					
001	Gli utenti devono poter scegliere la categoria delle domande.				
002	Ogni domanda deve avere una sola categoria.				

Requisito	Req-02	Priorità	1	Versione	1.0
Nome	Difficoltà domande				
Note	Le domande devono hanno tre livelli di difficoltà: facile, medio e difficile.				
Sotto requisiti					
001	Gli utenti devono poter scegliere la difficoltà delle domande.				
002	Le domande hanno tutte una risposta, anche le più difficili.				

Requisito	Req-03	Priorità	1	Versione	1.0
Nome	Risposte				
Note	L'applicativo deve avere un sistema di risposte dove ogni risposta esatta il giocatore ottiene un punteggio.				
Sotto requisiti					
001	Per ogni risposta corretta si ottiene un punteggio differente definito dalla difficoltà della domanda.				
002	Devono esserci due tipi di risposte: multiple, con una sola risposta e vero o falso.				

Requisito	Req-04	Priorità	1	Versione	1.0
Nome	Ranking				
Note	L'applicativo deve avere un sistema di classifica basato sui punteggi dei giocatori.				
Sotto requisiti					
001	Deve esserci una classifica che tiene conto dei punteggi totali di tutti i giocatori.				
002	I giocatori devono poter vedere la propria posizione in classifica alla fine della partita.				

Requisito	Req-05	Priorità	1	Versione	1.0
Nome	Profilo giocatore				
Note	L'applicativo deve avere una pagina dove l'utente deve poter inserire nome e rimuovere un giocatore, poter aggiungere un numero giocatori con un massimo di 10.				
Sotto requisiti					
001	Deve esserci una funzione che aggiorna i punteggi e le statistiche dei giocatori.				

Spiegazione elementi tabella dei requisiti:

ID: identificativo univoco del requisito.

Nome: breve descrizione del requisito.

Priorità: indica l'importanza di un requisito nell'insieme del progetto.

Versione: indica la versione del requisito.

Note: eventuali osservazioni importanti o riferimenti ad altri requisiti.

Sotto requisiti: elementi che compongono il requisito.

3.3 Use case

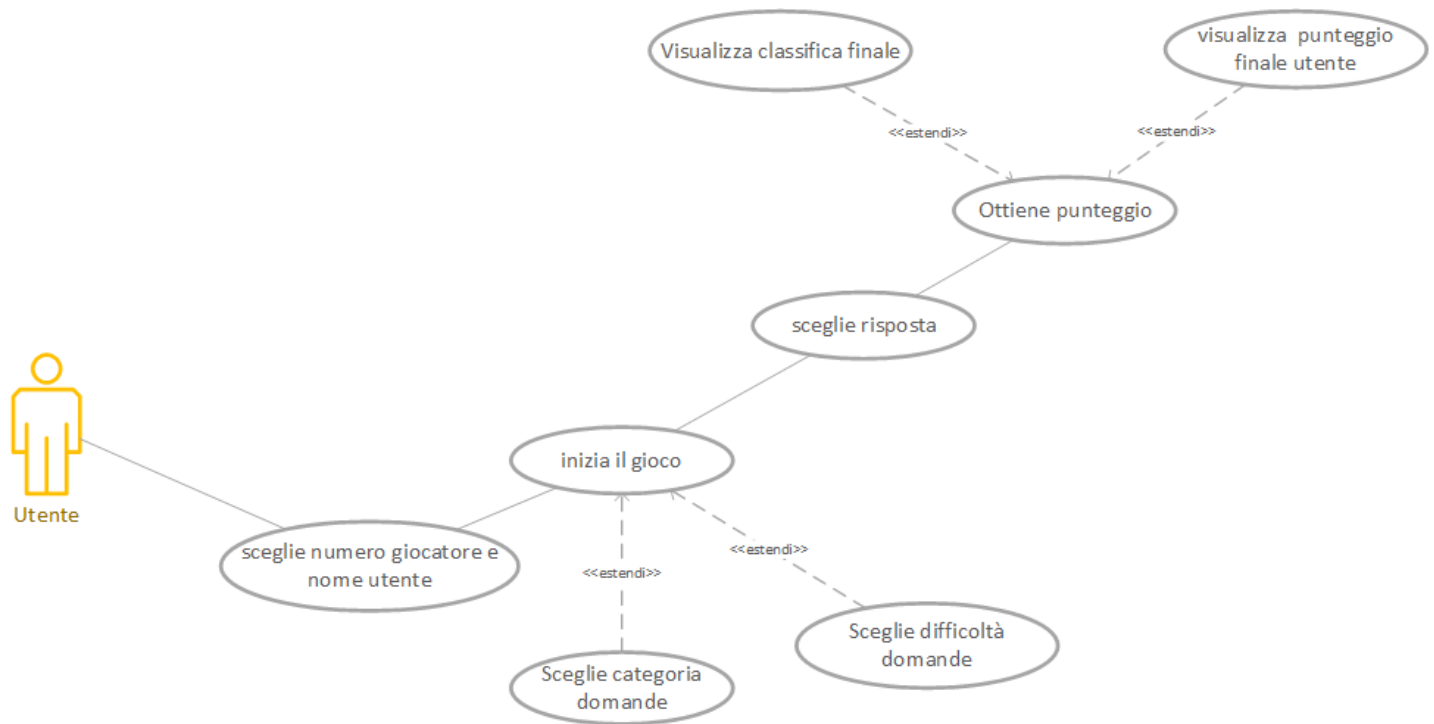


Figura 1 Use Case

Questo Use case mostra come l'utente può scegliere in quante persone giocare e che tipo di domande si vuole affrontare nel quiz, dove dopo aver iniziato il gioco, risponderà alle domande. Ogni risposta può assegnare un punteggio all'utente a dipendenza della risposta. Dopo aver terminato le domande verrà mostrato il punteggio delle persone e la classifica.

3.4 Pianificazione

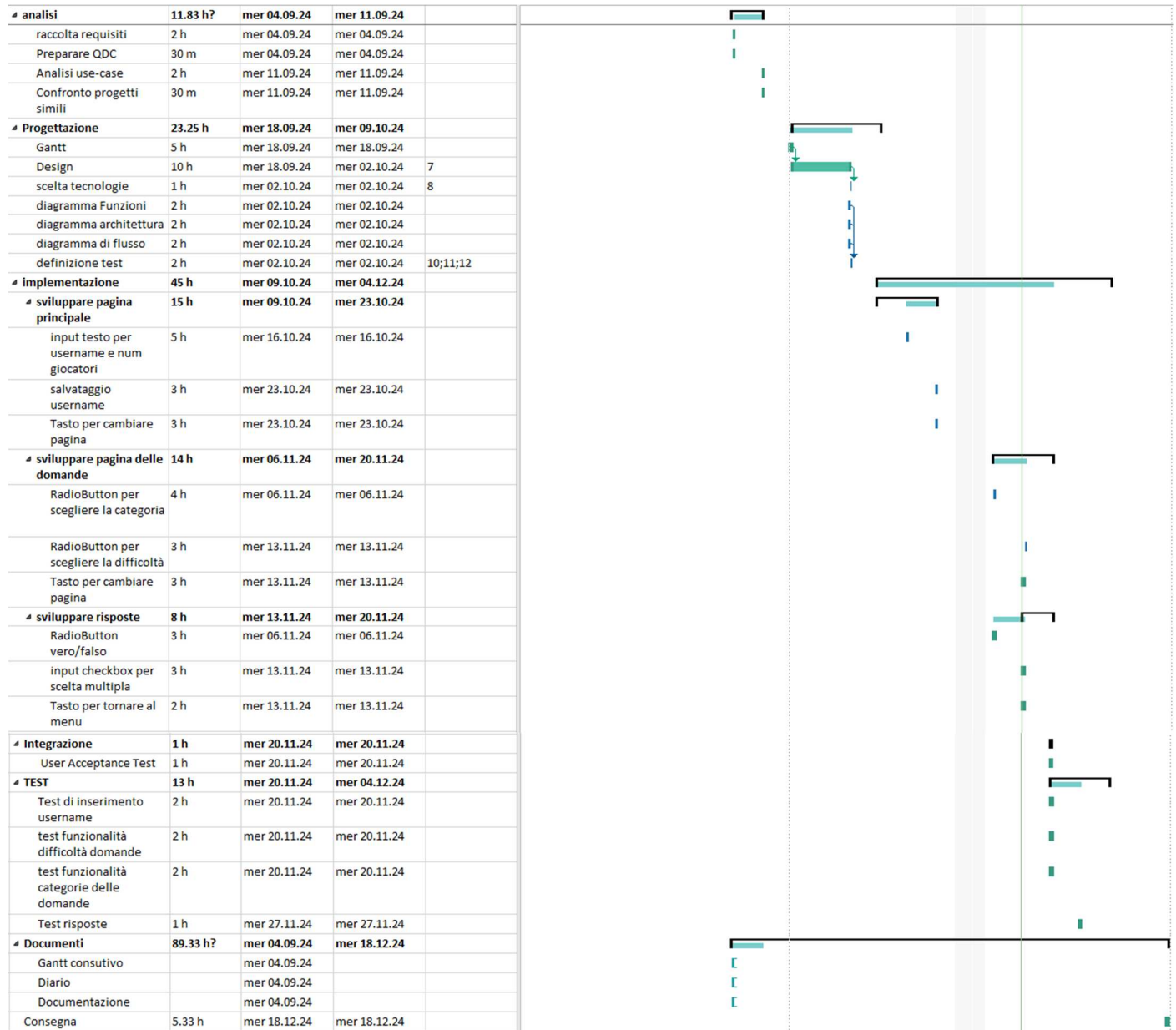



Figura 2 Pianificazione Gantt preventivo

Questo è il Gantt preventivo. Per il progetto sono previste molte ore per l'implementazione poiché viene usato Java Swing che si deve imparare ad usare.

	SAMT – Sezione Informatica	Pagina 10 di 31
	Documentazione Quiz Game	

3.5 Analisi dei mezzi

Per realizzare questo progetto è servito un Computer, l'accesso a Internet e un IDE di sviluppo.

3.5.1 Software

- NetBeans 23.0.1
- Microsoft Word 2019
- Microsoft Visio 2019
- Microsoft Project 2019

3.5.2 Librerie

- javax.swing
- PlantUML.jar
- gson-2.11.0.jar

3.5.3 Hardware

Per l'hardware è stato utilizzato solo il Computer fornito dalla scuola, con le seguenti caratteristiche:

Versione 22H2 (build SO 19045.4780)
 Windows: Windows 10 Education
 RAM: 32 GB
 CPU: Intel® Core™ i7-13700 CPU @ 2.10 GHz
 GPU: NVIDIA T400 4GB

4 Progettazione

4.1 Design dell'architettura del sistema

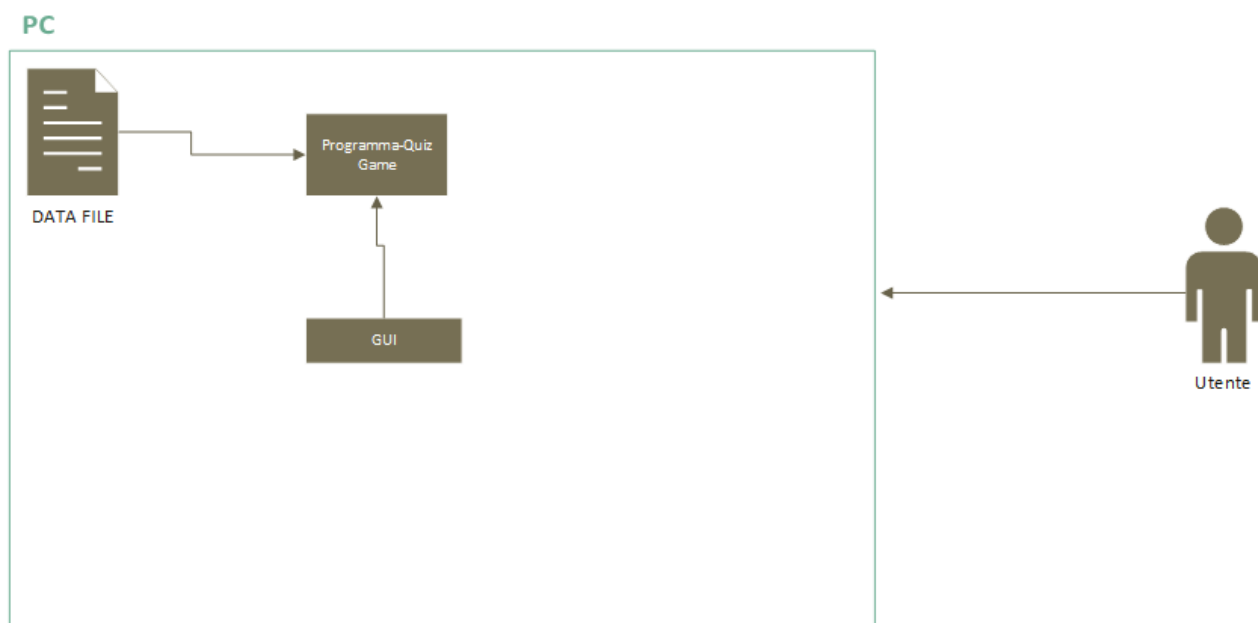



Figura 3 Diagramma Architettura

Questo diagramma mostra come l'utente interagisce con il Computer quando il programma viene avviato, l'utente interagisce con il programma tramite una GUI, il programma usa un Data File che contiene informazioni utili al gioco.

	SAMT – Sezione Informatica	Pagina 12 di 31
	Documentazione Quiz Game	

4.2 Design delle interfacce

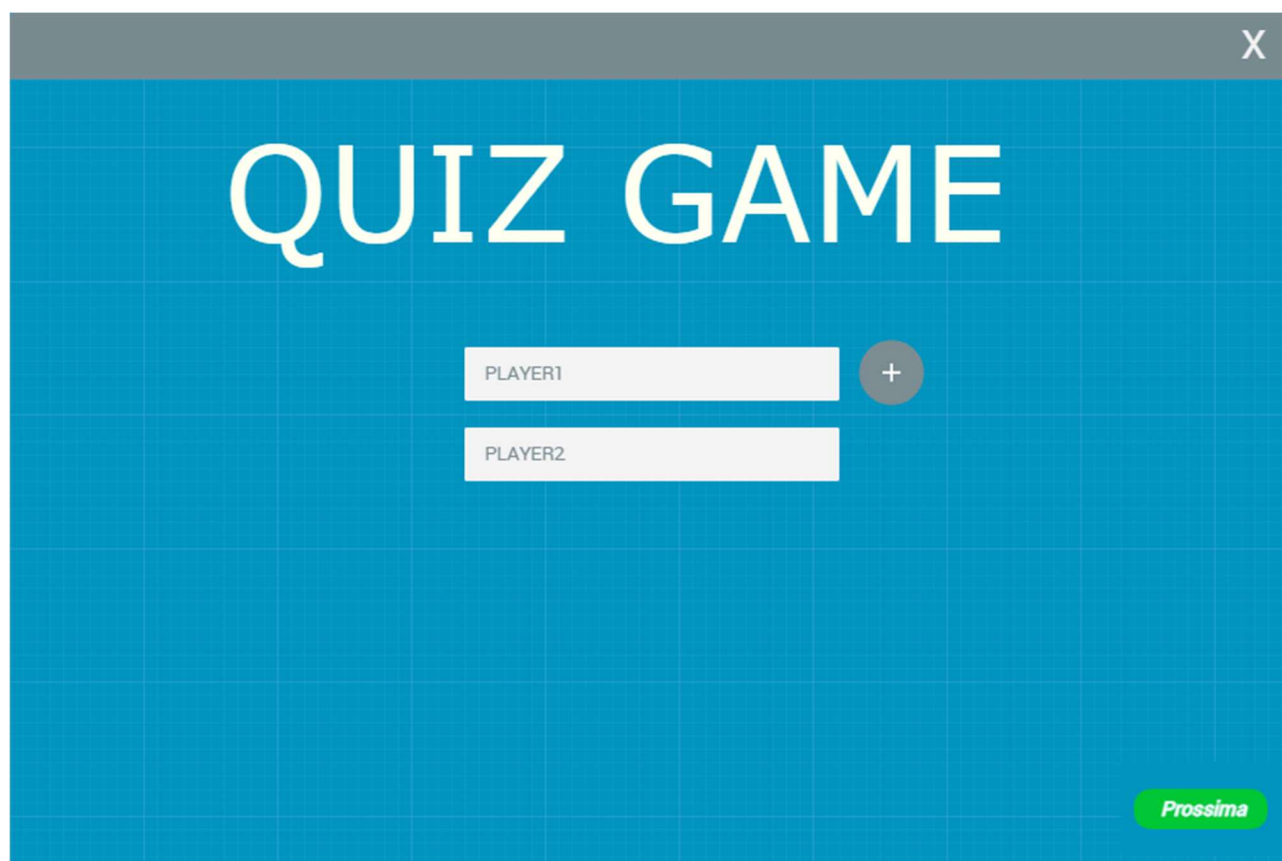


Figura 4 Pagina iniziale Quiz Game

Questa interfaccia è la pagina iniziale del Quiz Game dove si inserisce il numero dei giocatori e il loro nomi. L'utente potrà cambiare pagina con “prossima”.



	SAMT – Sezione Informatica	Pagina 13 di 31
	Documentazione Quiz Game	



Figura 5 Pagina impostazioni domande Quiz Game

In questa pagina l'utente potrà scegliere le caratteristiche delle domande dove in seguito deve rispondere, potendo scegliere difficoltà e la categoria, in fondo c'è la possibilità di scegliere se andare alla pagina successiva o alla pagina iniziale.

	SAMT – Sezione Informatica	Pagina 14 di 31
	Documentazione Quiz Game	



The screenshot shows a quiz interface with a blue grid background. At the top right, there is a close button 'X' and a question counter '1/20'. The text 'PLAYER 1 risponde' is centered at the top. The question is 'La bandiera rossa con una croce in mezzo a quale nazione corrisponde?'. Below the question are four radio button options: SVIZZERA, SVEZIA, GERMANIA, and ITALIA. At the bottom left is an orange button labeled 'PAGINA INIZIALE' and at the bottom right is a green button labeled 'Prossima'.

Figura 6 Pagina domanda/risposta Quiz Game

In questa pagina viene scelta la risposta dall'utente che può essere con 4 scelte oppure può essere un vero o falso. In alto a destra il numero della domanda, a quale domanda siamo, in alto al centro viene specificato l'utente che deve rispondere, subito sotto la domanda e le risposte e in fondo c'è la possibilità di scegliere se andare alla pagina successiva o alla pagina iniziale.

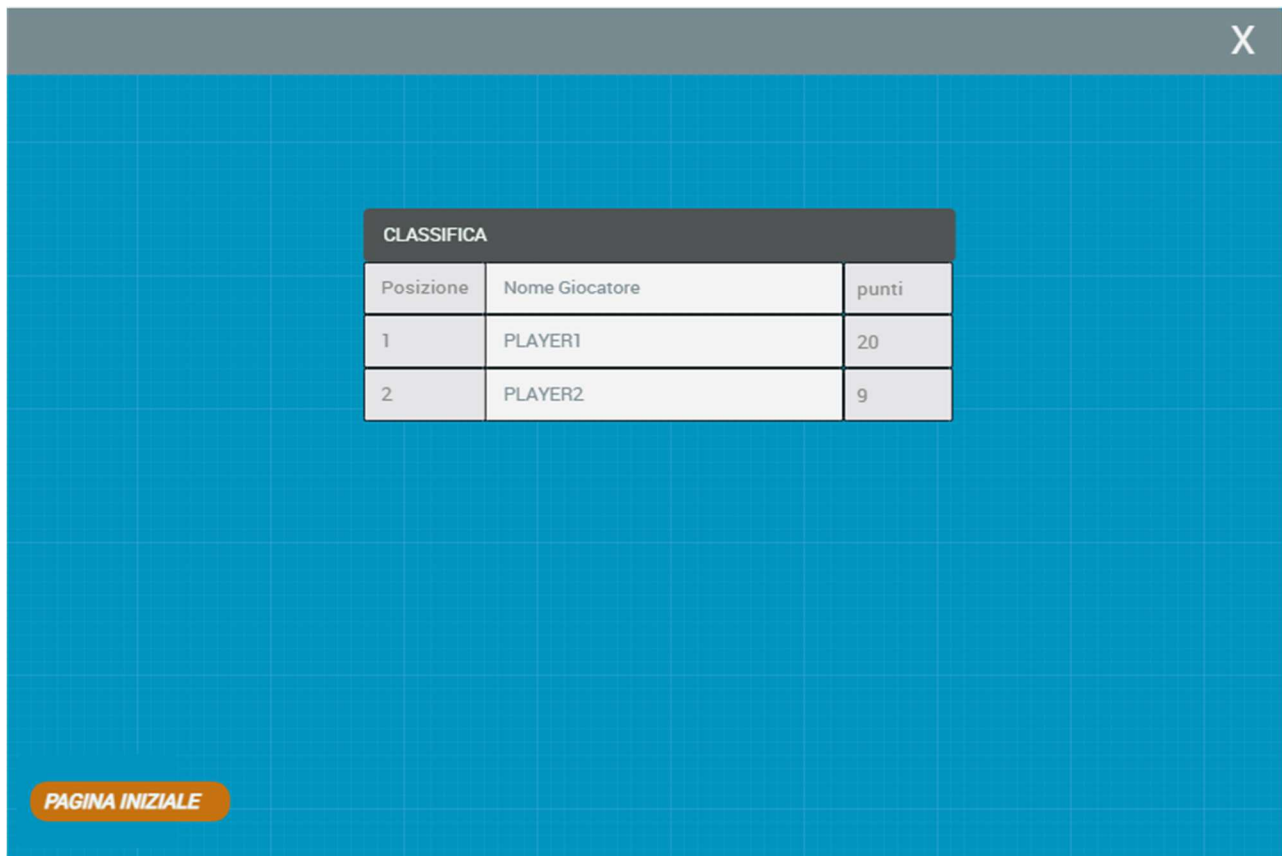


Figura 7 Pagina punteggio e classifica Quiz Game

Questa è l'ultima pagina che mostra la classifica dei giocatori e i punteggi delle varie persone che hanno giocato, viene mostrata una classifica. L'utente può tornare alla pagina iniziale oppure chiudere l'applicazione.

4.3 Design procedurale

4.3.1 Diagramma di flusso

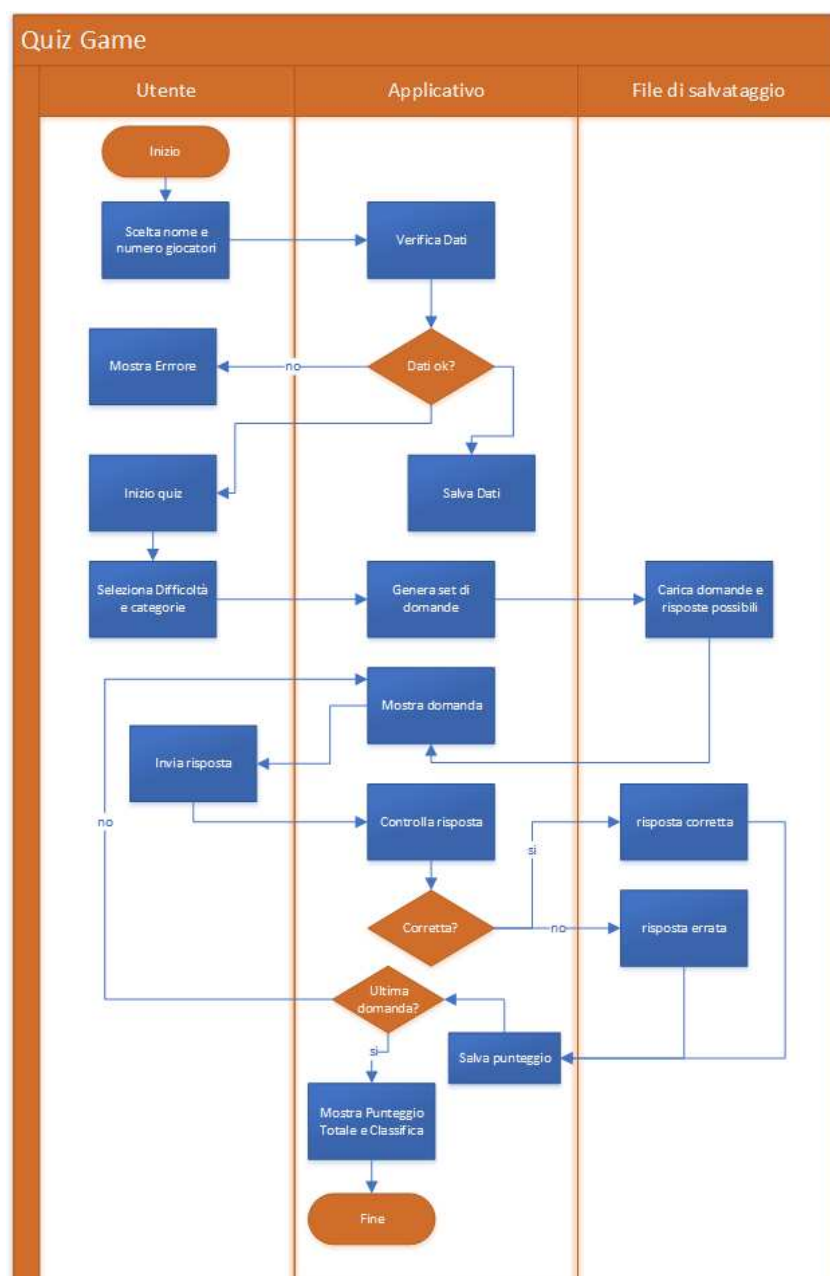


Figura 7 Design Procedurale Quiz Game

Questo diagramma mostra l'iterazione tra utente, applicativo e dove vengono memorizzati i dati, il file di salvataggio. L'utente inserisce nome e per ogni giocatore che gioca, sceglie le caratteristiche delle domande e infine visualizza il punteggio. L'applicativo controlla i dati inseriti dall'utente, verifica le risposte e salva i punteggi degli utenti. Il file di salvataggio carica le domande e le risposte.

4.3.2 Diagramma delle classi

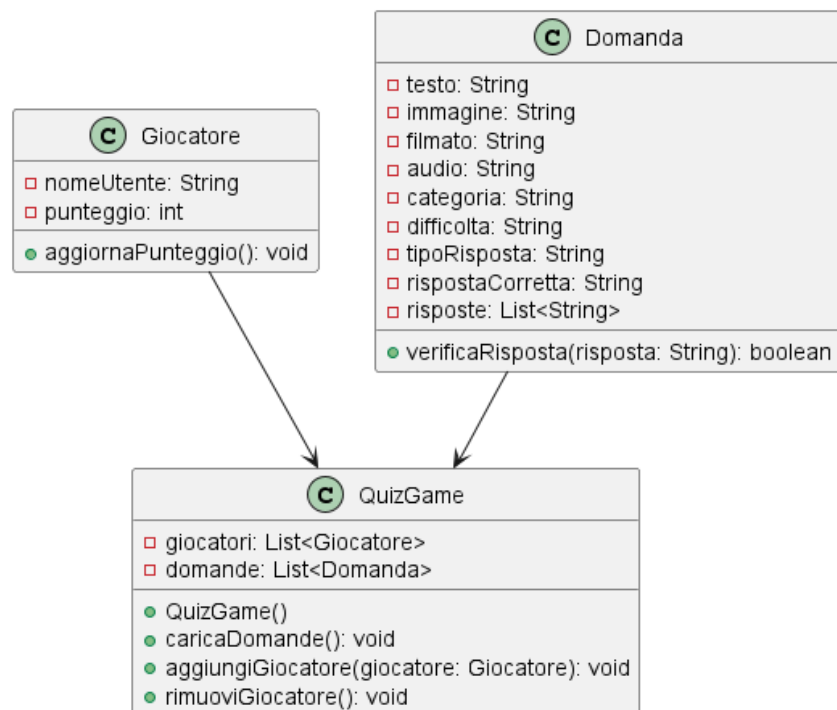


Figura 8 Diagramma delle classi e funzioni

Il diagramma delle classi mostra le 3 classi del gioco. QuizGame, che è dove vengono visualizzate le domande e la classifica, dentro ci sono anche le domande. La classe Domanda, è presente la struttura con variabili per generare la domanda. Poi c'è la classe Giocatore che gli viene assegnato il nome dell'utente e il punteggio.

5 Implementazione

5.1 Struttura cartelle

Una cartella principale che contiene le cartelle generate da NetBeans, mentre sotto src c'è una cartella quizgame dove c'è dentro:

- Cartella UI, contiene le diverse interfacce del programma:
 - Classifica.java
 - DomandaMultipla.java
 - DomandaVeroFalso.java
 - Impostazioni.java
 - MenuIniziale.java
- Domanda.java, sono le singole domande.
- Giocatore.java, sono i singoli giocatori, dove viene verificato il nome e il punteggio di ogni giocatore.
- Principale.java, il template delle varie UI.
- gson-2.11.0.jar, serve a Java per implementare e gestire file JSON
- quiz.json, contiene le informazioni delle domande.
- QuizGame.java, è il gestore centrale, dove vengono prese le domande da Quiz.txt e vengono fatte le liste di giocatori e domande.

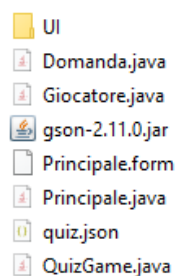



Figura 9 Struttura applicativo

5.2 Classi Principali

5.2.1 Giocatore

La classe Giocatore, che eredita dalla classe QuizGame, si occupa di tenere traccia dei partecipanti. Ogni giocatore ha un nome utente e un punteggio che aumenta durante il gioco. Se il giocatore non inserisce un nome, viene generato automaticamente un nome casuale tramite le lettere dell'alfabeto.

	SAMT – Sezione Informatica	Pagina 19 di 31
	Documentazione Quiz Game	

5.2.2 Domanda

La classe Domanda, che eredita dalla classe QuizGame, rappresenta la struttura delle domande utilizzate nel gioco. Ogni domanda è caratterizzata da un testo, una categoria (es. Sport, Storia), una difficoltà (Facile, Medio, Difficile) e la risposta corretta. Nel caso delle domande a scelta multipla, ci sono anche le opzioni di risposta. Per verificare le risposte viene utilizzato il metodo verificaRisposta, che permette di verificare la risposta alla domanda.

5.2.3 QuizGame

QuizGame è la classe centrale che gestisce tutto il gioco. Dove vengono caricati i dati delle domande da un file JSON esterno. La classe tiene anche traccia della lista dei giocatori, c'è anche la possibilità di aggiungere e rimuovere partecipanti. La funzionalità principale del metodo che permette di caricare e leggere un file JSON, con dentro l'array dove ci sono diversi dizionari con chiave e valore. Ogni chiave viene verificata se esiste, se questa chiave c'è, viene preso il valore, nel caso ci sia anche la chiave risposte, vengono prese anche quelle. Dopo aver preso i valori si controlla in java che questi valori presi dalle chiavi effettivamente esistono, tranne la List di risposte, che serve solo alle domande con scelta multipla. In seguito, nel caso passino il controllo, viene creata una nuova domanda dove viene aggiunta alla lista delle domande. In caso generi qualche eccezione, la prende e la mostra.

```
public void caricaDomande() {
    try {
        File file = new File(System.getProperty("user.dir") + "/src/quizgame/quiz.json");


        Gson gson = new Gson();
        try (FileReader reader = new FileReader(file)) {
            JSONArray jsonArray = gson.fromJson(reader, JSONArray.class);

            for (JsonElement element : jsonArray) {
                JsonObject jsonObject = element.getAsJsonObject();

                String testo = jsonObject.has("testo") ? jsonObject.get("testo").getString() : null;
                String categoria = jsonObject.has("categoria") ? jsonObject.get("categoria").getString() : null;
                String difficolta = jsonObject.has("difficolta") ? jsonObject.get("difficolta").getString() : null;
                String tipo = jsonObject.has("tipo") ? jsonObject.get("tipo").getString() : null;
                String rispostaCorretta = jsonObject.has("rispostaCorretta") ?
                jsonObject.get("rispostaCorretta").getString() : null;

                List<String> risposte = new ArrayList<>();
                if (jsonObject.has("risposte")) {
                    JSONArray risposteArray = jsonObject.getAsJsonArray("risposte");
                    for (JsonElement risposta : risposteArray) {
                        risposte.add(risposta.getString());
                    }
                }

                if ((testo != null && !testo.isEmpty()) && (tipo != null && !tipo.isEmpty()) && (difficolta != null &&
                !difficolta.isEmpty()) && (rispostaCorretta != null && !rispostaCorretta.isEmpty()) && (categoria != null &&
                !categoria.isEmpty())) {
```

	SAMT – Sezione Informatica	Pagina 20 di 31
	Documentazione Quiz Game	

```

Domanda domanda = new Domanda(testo, null, null, null, categoria, difficolta, tipo, rispostaCorretta,
risposte);
    domande.add(domanda);
    }
    }
    }
} catch (Exception e) {
    System.err.println("Errore durante il caricamento delle domande: " + e.getMessage());
}

```

5.3 Interfaccia Grafica


L'interfaccia utente è stata realizzata utilizzando Java Swing. Per rendere la struttura più ordinata, ogni pagina del gioco è stata implementata come un pannello separato.

5.3.1 Principale

Nel principale vengono gestite le varie pagine, richiama le varie interfacce grafiche, i metodi per verificare le domande per esempio, inoltre è usato anche come template per le altre pagine. Questa classe ha altri due metodi importanti, richiamati da due Button, che servono a far andare avanti e indietro le diverse pagine, dove gli viene applicato a entrambi un Thread sleep per rendere più lento il cambiamento di pagina in modo che l'utente non vada troppo veloce, uno di questi è prossimaActionPerformed, grazie a questo metodo le pagine vanno avanti, dove dentro richiamano le altre pagine e in seguito tolgono la pagina attuale e mettono quella nuova. L'altro metodo invece è precedenteActionPerformed, che ritorna da qualsiasi pagina alla pagina del menu iniziale.

5.3.2 Menu Iniziale

Il menu iniziale consente ai giocatori di inserire i propri nomi tramite dei TextField e di salvare i nuovi giocatori. È possibile aggiungere fino a dieci giocatori e toglierne fino al minimo di uno. Se il numero massimo viene raggiunto, il Button per aggiungere nuovi giocatori viene disabilitato. Se il numero minimo viene raggiunto il Button per rimuovere i giocatori viene disabilitato. Per salvare i giocatori serve il metodo inviaNomi, che inizialmente rimuove tutti i giocatori se sono state fatte partite precedentemente, poi crea due array, uno che è dove l'utente inserisce il nome dei vari giocatori, l'altro invece serve a creare un nuovo array di giocatori. In seguito viene creato un ciclo dove a ogni JTextField abilitata dall'utente, viene assegnato il nome del giocatore, con il punteggio a zero, poi viene usato il set che controlla che il nome sia stato inserito, altrimenti ne genera uno lui in modo casuale con cinque lettere dell'alfabeto. Infine aggiunge alla lista dei giocatori del quizgame il nuovo giocatore.

	SAMT – Sezione Informatica	Pagina 21 di 31
	Documentazione Quiz Game	

```

public void inviaNomi() {
    quizGame.rimuoviGiocatore();
    JTextField fields[] = {nomeUtente1, nomeUtente2, nomeUtente3, nomeUtente4, nomeUtente5, nomeUtente6,
nomeUtente7, nomeUtente8, nomeUtente9, nomeUtente10};
    Giocatore utenti[] = new Giocatore[fields.length];
    for (int i = 0; i < fields.length; i++) {
        if (fields[i].isEnabled()) {
            utenti[i] = new Giocatore(fields[i].getText(), 0);
            utenti[i].setNomeUtente(fields[i].getText());
            quizGame.aggiungiGiocatore(utenti[i]);
        }
    }
}

```

5.4 Impostazioni

In questa classe l'utente seleziona la difficoltà (facile, medio e difficile) e la categoria (Sport, Materie scolastiche, Storia, Geografia, Lingue), che grazie al `getCategoria` e `getDifficolta` prendono la categoria e la difficoltà inserite dall'utente.


5.5 Domande

Nel gioco ci sono due tipi di domande, quelle con la risposta a scelta multipla e quelle vero/falso, in entrambe le classi c'è la scelta di chi deve rispondere, che viene scelto in sequenza con il metodo `scegliGiocatore`, dove prende la lista dei giocatori, gli viene passato il numero del giocatore che deve rispondere alla domanda, in seguito viene messo il nome nella pagina per far vedere chi deve rispondere e lo inserisce come valore nella variabile `giocatore`, per impostare il valore corrente.

5.5.1 Vero/Falso

In questo tipo di domande ci sono solo due opzioni, vero o falso. Un `radioButton` corrisponde a vero mentre l'altro a falso. Per verificare la risposta viene usato il metodo `verificaRisposta`, dove se la risposta è giusta viene aggiornato il punteggio del giocatore che ha risposto correttamente, mentre se sbaglia non aggiunge i punti.

Per scegliere la domanda viene usato il metodo `scegliDomanda` che inizialmente gli viene passato il numero della domanda attuale. Questo metodo controlla se la categoria e la difficoltà delle domande sono le stesse di quelle scelte dall'utente, tutto questo grazie a un ciclo che controlla domanda per domanda e nel caso sia una domanda con le caratteristiche richieste dall'utente, la domanda viene aggiunta in una lista dove ci saranno altre domande validate e in seguito verranno scelte in modo casuale fra quelle inserite nella lista. Infine viene impostato il testo della domanda nella pagina e viene anche presa la risposta corretta della domanda scelta in modo casuale tra la lista delle domande valide.

	SAMT – Sezione Informatica	Pagina 22 di 31
	Documentazione Quiz Game	

```

public void scegliDomanda(int numeroDomanda) {
    String cat = impo1.getCategoria();
    String diff = impo1.getDifficolta();
    contentDomanda.setText("");
    numDomanda.setText(Integer.toString(numeroDomanda));
    quizGame.QuizGame();
    List<Domanda> domande = quizGame.getDomande();
    List<Domanda> domandeUtili = new ArrayList<>();
    int contaDomande = 0;
    for (int i = 0; i < domande.size(); i++) {
        if ((domande.get(i).getTipoRisposta().equals("VF")) && (domande.get(i).getCategoria().equals(cat)) &&
(domande.get(i).getDifficolta().equals(diff))) {
            domandeUtili.add(domande.get(i));
            contaDomande++;
        }
    }

    if (contaDomande > 0) {
        Random random = new Random();
        int indiceCasuale = random.nextInt(contaDomande);
        Domanda domandaCasuale = domandeUtili.get(indiceCasuale);
        contentDomanda.setText(domandaCasuale.getTesto());
        rispostaCorretta = domandaCasuale.getRispostaCorretta();
    }
}

```


5.5.2 Scelta Multipla

In questo tipo di domande ci sono solo quattro opzioni, che cambiano in base alla domanda. Ogni radioButton corrisponde a una domanda diversa. Per verificare la risposta viene usato il metodo verificaRisposta, dove se la risposta è giusta viene aggiornato il punteggio del giocatore che ha risposto correttamente, mentre se sbaglia non aggiunge i punti.

Per scegliere la domanda viene usato il metodo scegliDomanda, che è simile a quello usato per le domande vero/falso. Inizialmente gli viene passato il numero della domanda attuale. Questo metodo controlla se la categoria e la difficoltà delle domande sono le stesse di quelle scelte dall'utente, tutto questo grazie a un ciclo che controlla domanda per domanda e nel caso sia una domanda con le caratteristiche richieste dall'utente, la domanda viene aggiunta in una lista dove ci saranno altre domande validate e in seguito verranno scelte in modo casuale fra quelle inserite nella lista. Infine viene impostato il testo della domanda nella pagina, poi carica le quattro possibili risposte nei quattro radioButton e in seguito viene anche presa la risposta corretta della domanda scelta in modo casuale tra la lista delle domande valide.

5.6 Classifica

La classifica finale ordina i giocatori in base ai loro punteggi e mostra il risultato di ogni giocatore. Questa pagina viene visualizzata alla fine del gioco. Il metodo per ordinare la classifica è aggiornaClassifica, che grazie ai metodi sort e compare ordina i giocatori in base al punteggio.

	SAMT – Sezione Informatica	Pagina 23 di 31
	Documentazione Quiz Game	

5.7 quiz.json

Il quiz.json ha due tipi di struttura, una per il vero/falso mentre l'altro per le domande a scelta multipla. Tutto viene inserito in un'array, dove dentro questo array c'è un dizionario per ogni domanda, che è strutturata da chiave e valore

5.7.1 Contenitore vero/falso

La struttura del json per le domande vero/falso è la seguente:

- **testo:** Il testo della domanda,
- **categoria:** La categoria della domanda (es. Sport),
- **difficoltà:** La difficoltà della domanda (Facile, Medio, Difficile),
- **tipo:** Il tipo di domanda (VF che sarebbe Vero/Falso e Multipla),
- **rispostaCorretta:** La risposta corretta alla domanda.

5.7.2 Contenitore scelta Multipla

La struttura del dizionario per le domande a scelta multipla, dove vengono aggiunte le risposte all'interno di un array, è la seguente:

- **testo:** Il testo della domanda,
- **categoria:** La categoria della domanda (es. Sport),
- **difficoltà:** La difficoltà della domanda (Facile, Medio, Difficile),
- **tipo:** Il tipo di domanda (VF che sarebbe Vero/Falso e Multipla),
- **rispostaCorretta:** La risposta corretta alla domanda,
- **risposte:** Le diverse risposte possibili.

6 Protocollo di test

Test Case	TC-001	Nome	Inserimento giocatore
Riferimento	REQ-05		
Descrizione	L'utente per giocare deve inserire il nome, nel caso contrario dovrà darlo il sistema in modo autonomo		
Prerequisiti	Sistema già pronto per rispettare questo problema e lista pronta per far inserire il nome dei diversi giocatori.		
Procedura	<ol style="list-style-type: none"> 1. L'utente inserisce il giocatore di nome «fedè» 2. L'utente inserisce il giocatore senza nome 		
Risultati attesi	Nel sistema, il giocatore «fedè» viene messo in una lista di giocatore, mentre quello che ha lasciato vuoto, il sistema gli genera un nome casuale.		

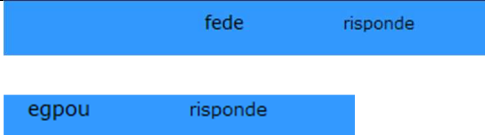


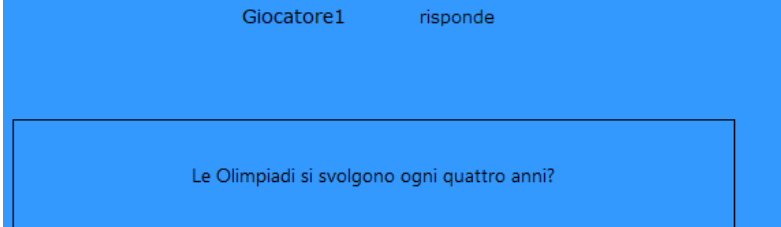
Test Case	TC-002	Nome	Rimozione giocatore
Riferimento	REQ-05		
Descrizione	L'utente per giocare deve poter rimuovere il nome, fino a un giocatore		
Prerequisiti	Sistema già pronto per rispettare questo problema e lista pronta per adattarsi ai giocatori eliminati.		
Procedura	<ol style="list-style-type: none"> 1. L'utente inserisce il giocatore di nome «Giocatore1» 2. L'utente inserisce il giocatore di nome «fedè» 3. L'utente rimuovere il giocatore «fedè» 		
Risultati attesi	Nel sistema, il giocatore «fedè» viene tolto dalla lista di giocatori, dunque sarà visibile il giocatore «Giocatore1» e il bottone per rimuovere il giocatore non si potrà più usare.		

Test Case	TC-003	Nome	Classifica con punteggi da migliore a peggiore
Riferimento	Req-03 Req-04		
Descrizione	La classifica deve essere in ordine decrescente, con nomi, posizioni e punteggio dei vari giocatori.		
Prerequisiti	Metodo con la lista dei nomi dei giocatori con i propri punteggi e un metodo per inserire la gente in classifica		
Procedura	<ol style="list-style-type: none"> 1. L'utente inserisce il giocatore di nome «Giocatore1» 2. L'utente inserisce il giocatore di nome «fedè» 3. I giocatori hanno risposto a tutte le domande 4. Alla fine del gioco viene mostrata la classifica 		
Risultati attesi	«Giocatore1» ha risposto correttamente a più domande, quindi dovrebbe essere primo mentre «fedè» secondo.		

Test Case	TC-004	Nome	Possibilità di tornare al menù iniziale
Riferimento			
Descrizione	L'utente deve avere la possibilità di tornare alla pagina di inizio dove si inseriscono i giocatori		
Prerequisiti	Button che permette di disabilitare la pagina attuale e di tornare alla pagina iniziale.		
Procedura	<ol style="list-style-type: none"> 1. L'utente inserisce il giocatore di nome «Giocatore1» 2. L'utente usa il Button «pagina iniziale» per tornare all'inizio 		
Risultati attesi	«Giocatore1» si ritrova alla pagina iniziale con lui come giocatore		

Test Case	TC-005	Nome	Creazione corretta delle domande
Riferimento	Req-01 Req-02		
Descrizione	Quando l'utente sceglie difficoltà e la categoria delle domande, le domande devono essere caricate correttamente.		
Prerequisiti	File contenente le domande, risposte, categoria, difficoltà,		
Procedura	1. L'utente seleziona «Sport» come categoria e «Facile» come difficoltà.		
Risultati attesi	Le domande devono appartenere al mondo dello sport e devono essere semplici da rispondere.		

6.1 Risultati test

Test Case	Risultato ottenuto	Stato									
TC-001		Passato									
TC-002		Passato									
TC-003	 <table border="1"> <thead> <tr> <th>Posizione</th><th>Nome Giocatore</th><th>Punti</th></tr> </thead> <tbody> <tr> <td>1</td><td>Giocatore1</td><td>30</td></tr> <tr> <td>2</td><td>federe</td><td>20</td></tr> </tbody> </table>	Posizione	Nome Giocatore	Punti	1	Giocatore1	30	2	federe	20	Passato
Posizione	Nome Giocatore	Punti									
1	Giocatore1	30									
2	federe	20									
TC-004	Utente tornato alla pagina iniziale, provato si dalle impostazioni delle domande, sia alle pagine delle domande sia alla classifica.	Passato									
TC-005		Passato									

6.2 Mancanze/limitazioni conosciute

Nel progetto mi sono accorto che se avessi dovuto implementare anche immagini, audio e filmati non avrei finito entro la consegna, quindi ho deciso di non implementarli. Ci sono stati problemi anche con l'accento delle lettere se avviato tramite jar.

7 Consuntivo

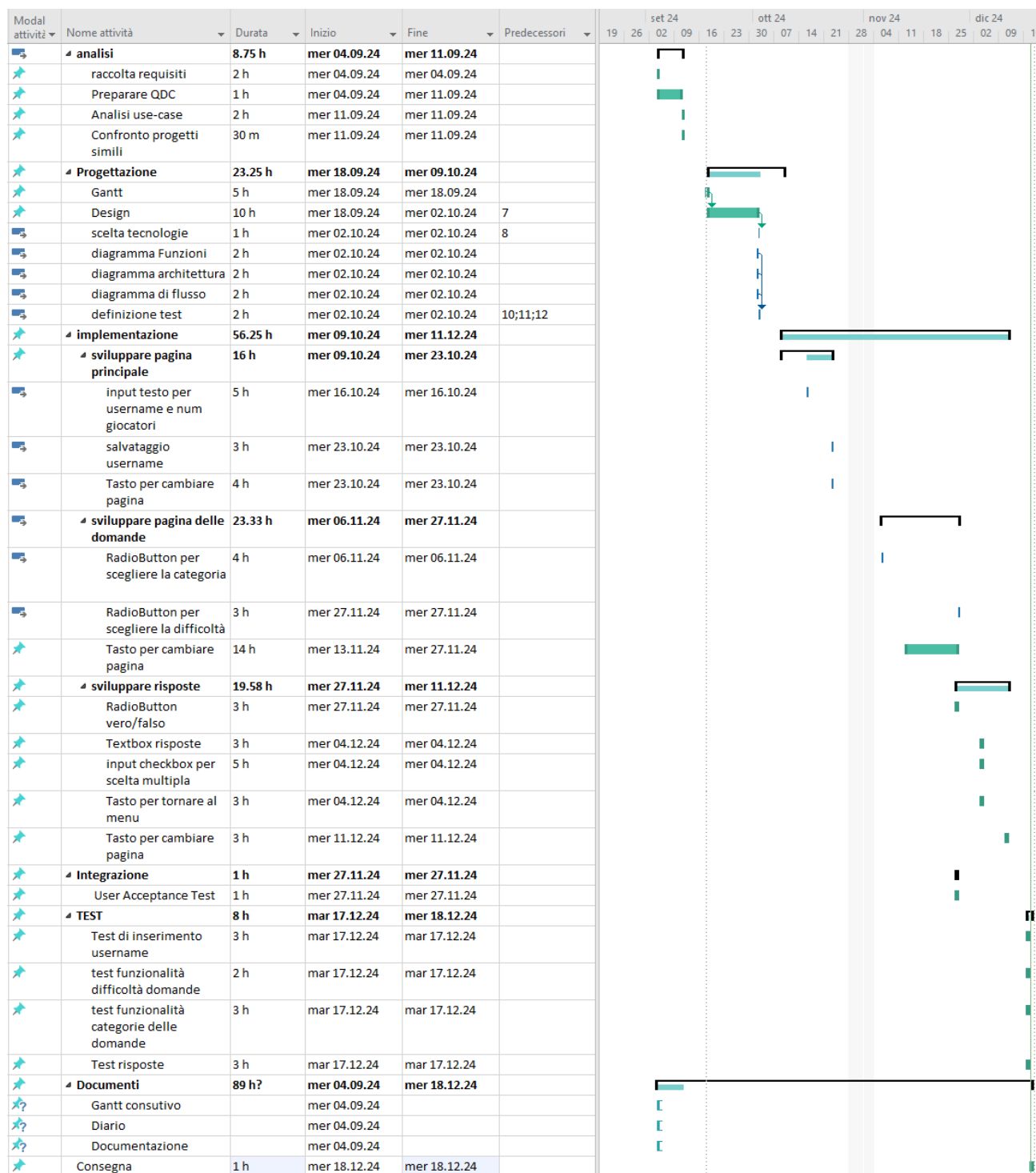



Figura 10 Gantt Consuntivo

Rispetto alla pianificazione il tempo dove ho impiegato qualche ora in più è stata l'implementazione.

	SAMT – Sezione Informatica	Pagina 29 di 31
	Documentazione Quiz Game	

8 Conclusioni


Questo progetto essendo che ce ne sono diversi nel mondo, anche con più funzionalità e più belli, a questo progetto potrebbero essere aggiunte nuove funzionalità, del lavoro che ho svolto fino al termine del progetto sono abbastanza soddisfatto. L'applicativo anche se ne esistono diversi simili, potrebbe sempre essere utilizzato per giocare con gli amici o come passatempo divertente per imparare nuove cose in compagnia.

8.1 Sviluppi futuri

Se dovessi riprendere questo progetto in un futuro, sono certo che implementerei le domande con filmato, audio e immagini, per rendere il gioco più intrattenente. Mentre a questione di pianificazione a tenere meglio il conto del tempo, che sembra tanto ma alla fine è minore di quello previsto.

8.2 Considerazioni personali

In questo progetto ho imparato che se pianifico una cosa, il tempo sarà sempre maggiore di quello pensato poiché ci saranno sempre problemi nel percorso del lavoro, per risolverli serve tempo, il tempo che bisognerebbe impiegare per le altre cose, altrimenti non lo finirai mai il progetto o comunque non lo completerai al 100%. Inoltre ho compreso la maggior parte delle basi per sviluppare un progetto da zero. I risultati che ho ottenuto alla fine sono all'incirca quelli che mi aspettavo, anche se con qualche difficoltà ogni tanto.

	SAMT – Sezione Informatica	Pagina 30 di 31
	Documentazione Quiz Game	

9 Bibliografia

9.1 Sitografia

1. <https://www.youtube.com/watch?v=qHBY9LpxJk0>, 09-10-2024
2. <https://forum.html.it/forum/showthread.php?threadid=1474142>, 09-10-2024
3. <https://docs.oracle.com/javase/10/tools/javapackager.htm#JSWOR719>, 09-10-2024
4. <https://www.bing.com/videos/riverview/relatedvideo?&q=fare+un+jform+nebeans+&&mid=71D4097C27276100026671D4097C272761000266&mmscn=mtsc&aps=742&FORM=VRDGAR>, 09-10-2024
5. <https://www.tutorialspoint.com/generate-random-boolean-in-java>, 27-11-2024
6. <https://stackoverflow.com/questions/10709803/java-comparator-how-to-sort-by-integer>, 27-11-2024
7. <https://www.iprogrammatori.it/articoli/java/comparazione-ordinamento-degli-oggetti-java>, 27-11-2024

10 Glossario

Termine	Significato
GUI	Interfaccia grafica
Java	Linguaggio di programmazione orientata agli oggetti
jar	File di archivio contenente classi Java e altre risorse necessarie all'esecuzione di applicazioni
Ranking	Classifica

11 Indice delle figure

Figura 1 Use Case	8
Figura 2 Pianificazione Gantt preventivo.....	9
Figura 3 Diagramma Architettura.....	11
Figura 4 Pagina iniziale Quiz Game	12
Figura 5 Pagina impostazioni domande Quiz Game	13
Figura 6 Pagina domanda/risposta Quiz Game	14
Figura 7 Design Procedurale Quiz Game	16
Figura 8 Diagramma delle classi e funzioni.....	17
Figura 9 Struttura applicativo.....	18
Figura 10 Gantt Consuntivo	28

12 Allegati

Elenco degli allegati:

- Tecnologie
- Requisiti
- Use Case
- UI
- Gantt consuntivo e preventivo
- Diagramma delle Classi
- Diagramma di Architettura
- Diagramma di Flusso