Quiz Game

# Indice

Sommario

[1 Indice 2](#_Toc185425640)

[2 Introduzione 3](#_Toc185425641)

[2.1 Informazioni sul progetto 3](#_Toc185425642)

[2.2 Abstract 3](#_Toc185425643)

[2.3 Scopo 3](#_Toc185425644)

[3 Analisi 4](#_Toc185425645)

[3.1 Analisi del dominio 4](#_Toc185425646)

[3.2 Analisi e specifica dei requisiti 4](#_Toc185425647)

[3.3 Use case 10](#_Toc185425648)

[3.4 Pianificazione 11](#_Toc185425649)

[3.5 Analisi dei mezzi 13](#_Toc185425650)

[3.5.1 Software 13](#_Toc185425651)

[3.5.2 Hardware 13](#_Toc185425652)

[4 Progettazione 13](#_Toc185425653)

[4.1 Design dell’architettura del sistema 13](#_Toc185425654)

[4.2 Design delle interfacce 15](#_Toc185425655)

[4.3 Design procedurale 19](#_Toc185425656)

[4.3.1 Diagramma di flusso 19](#_Toc185425657)

[4.3.2 Diagramma delle classi 20](#_Toc185425658)

[5 Implementazione 21](#_Toc185425659)

[5.1 Classe Main 21](#_Toc185425660)

[5.2 Classe UsernamePanel 21](#_Toc185425661)

[5.3 Classe SettingsPanel 22](#_Toc185425662)

[5.4 Classe QuizPanel 22](#_Toc185425663)

[5.4.1 Funzione loadQuestions 23](#_Toc185425664)

[5.4.2 Funzione displayQuestions 24](#_Toc185425665)

[5.4.3 Funzione pointAnswer 27](#_Toc185425666)

[5.5 Classe ResultPanel 29](#_Toc185425667)

[5.6 Classe LeaderboardManager 29](#_Toc185425668)

[5.7 Classe DetailsPanel 30](#_Toc185425669)

[6 Test 31](#_Toc185425670)

[6.1 Protocollo di test 31](#_Toc185425671)

[6.2 Risultati test 34](#_Toc185425672)

[6.3 Mancanze/limitazioni conosciute 34](#_Toc185425673)

[7 Consuntivo 35](#_Toc185425674)

[7.1 Progettazione 35](#_Toc185425675)

[7.2 Implementazione 36](#_Toc185425676)

[7.3 Test 38](#_Toc185425677)

[8 Conclusioni 39](#_Toc185425678)

[8.1 Sviluppi futuri 39](#_Toc185425679)

[8.2 Considerazioni personali 39](#_Toc185425680)

[9 Bibliografia 40](#_Toc185425681)

[9.1 Sitografia 40](#_Toc185425682)

[10 Glossario 40](#_Toc185425683)

[11 Indice delle figure 40](#_Toc185425684)

[12 Allegati 41](#_Toc185425685)

# Introduzione

## Informazioni sul progetto

Allievo: Leonardo Sciara

Docente Responsabile: Geo Petrini

SAMT Sezione Informatica 3BB Modulo 306

Inizio: 04.09.2024

Termine: 18.12.2024

## Scopo

Il progetto Quiz Game ha come obiettivo lo sviluppo di un sistema di quiz utilizzando Java, con la possibilità di personalizzare il quiz, rispondere alle domande e osservare la classifica dei migliori giocatori.Lo scopo didattico di questo progetto è quello di creare il miglior applicativo possibile tramite le conoscenze apprese in questi tre anni di scuola professionale nei vari moduli di programmazione.

# Analisi

## Analisi del dominio

L’applicativo di questo progetto sarà simile a quelli che trovi online, ti permette di

* Giocare col tuo username
* Scegliere categorie di domande, livelli di difficoltà e quantità di domande.
* Rispondere a delle domande con una o più opzioni di risposta.
* Visualizzare i punteggi al termine del quiz.

Il sistema sarà gestito attraverso Java

## Analisi e specifica dei requisiti

Inserimento nome giocatore:

* Al caricamento nella pagina Username, l'utente deve inserire il proprio nome tramite un input text che viene salvato in un file txt
* Se l'utente non inserisce niente, il tasto “Impostazioni” non compare

Inserimento impostazioni:

* File:
* Dopo aver premuto “Impostazioni”, l’utente va nel file Impostazioni
* Il file contiene 4 tasti; "Categoria", “Difficoltà”, “Quantità Domande” e “Inizio Gioco”
* Se l'utente non ha selezionato una categoria, una difficoltà o la quantità di domande, “Inizia gioco” non appare
* Categoria:
* L’utente sceglie una tra 6 categorie “Sport”, “Storia”, “Cinema”, “Scienza”, “Musica” e “Matematica” tramite dei RadioButton
* Difficoltà:
* L'utente sceglie una tra 3 difficoltà "Facile", "Medio" e "Difficile" tramite uno slider
* Quantità domande:
* L’utente sceglie un numero tra “1” e “30” tramite uno slider

Inizio gioco:

* Quando l'utente preme su "Inizia il gioco", seleziona le quantità di domande e risposte dal file di tipo txt e poi inizia il quiz
* Le domande cambiano in base alla categoria e alla difficoltà scelta
* Ogni domanda ha più possibili risposte, delle quali una o più sono corrette.
* Domande e risposte possono contenere anche delle immagini

Inserimento risposte:

* Le riposte saranno sottoforma di checkbox, se con più risposte, e sottoforma di radiobutton, se con una risposta singola
* Per rispondere alle domande c’è un determinato tempo a disposizione, dopo questo tempo viene calcolato come risposta sbagliata
* A dipendenza della difficoltà, il punteggio di ogni risposta corretta è differente
* Se in una domanda ci sono più risposte corrette e l’utente non inserisce il numero giusto di risposte, o le inserisce, ma sbagliate, il punteggio viene calcolato in base al numero di risposte corrette date e al numero di risposte corrette possibili
* Se l’utente non ha selezionato al meno una risposta, il tasto “Visualizza Risposta”, che serve per verificare la correttezza della risposta, non appare
* Dopo aver premuto “Visualizza Risposta”, il testo del tasto cambierà in “Procedi”, che premendolo, passerà alla domanda successiva

Visualizzazione risultati:

* Al termine del quiz, l’utente può visualizzare la classifica dei 10 migliori giocatori e il suo punteggio totale totalizzato
* L’utente può scegliere il tasto “Visualizza Dettagli", che porta nella pagina Details dove ti dice i punteggi fatti per ogni domanda e le risposte corrette a quella domanda
* L’utente può scegliere il tasto "Nuova Partita", che riporta al file per selezionare categorie, difficoltà e quantità di domande diverse
* L’utente può scegliere il tasto "Nuovo Giocatore", che riporta allo Username file per inserire un nuovo giocatore
* L’utente può scegliere il tasto "Chiudi Gioco", che chiude il gioco

Visualizzazione dettagli:

* L’utente potrà visualizzare i dettagli di ogni domanda, potendo vedere i punti totalizzati in quella domanda
* l’utente può premere il tasto “Indietro”, che porta l’utente alla pagina dei risultati

|  |  |
| --- | --- |
| **ID: REQ-01** | |
| **Nome** | Inserimento nome giocatore |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | L’utente deve inserire il proprio nome |
| **Sotto requisiti** | |
| **001** | L’utente deve inserire il nome desiderato nell’input text |
| **002** | Dopo aver inserito il nome, si dovrà premere “Impostazioni” per passare alla pagina delle impostazioni |

|  |  |
| --- | --- |
| **ID: REQ-02** | |
| **Nome** | Inserimento Impostazioni |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | L’utente deve configurare la categoria, la difficoltà e la quantità delle domande |
| **Sotto requisiti** | |
| **001** | L’utente sceglie una categoria tra “Sport”, “Storia”, “Cinema”, “Scienza”, “Musica” e “Matematica” tramite un RadioButton |
| **002** | L’utente seleziona una difficoltà tra “Facile”, “Medio” e “Difficile” tramite uno slider |
| **003** | L’utente seleziona una quantità di domande tra “1” e “30” tramite uno slider |
| **004** | Dopo aver selezionato i campi, si dovrà premere “Inizia Gioco” per iniziare il gioco |

|  |  |
| --- | --- |
| **ID: REQ-03** | |
| **Nome** | Inizio del gioco |
| **Priorità** | 2 |
| **Versione** | 1.0 |
| **Note** | L’utente deve rispondere alle domande che li vengono proposte |
| **Sotto requisiti** | |
| **001** | Vengono caricate domande in modo casuale in base alla categoria e difficoltà scelte |
| **002** | Ogni domanda ha più risposte possibili, una o più corrette. Di queste risposte, quelle sbagliate vengono inserite casualmente |
| **003** | Ogni domanda e risposta possono includere immagini |

|  |  |
| --- | --- |
| **ID: REQ-04** | |
| **Nome** | Inserimento risposte |
| **Priorità** | 2 |
| **Versione** | 1.0 |
| **Note** | L’utente deve selezionare le risposte tramite checkbox o radiobutton |
| **Sotto requisiti** | |
| **001** | Le risposte sono in formato checkbox per più risposte e radiobutton per una risposta |
| **002** | L’utente ha 30 secondi per rispondere a ogni domanda e superando questo tempo, la risposta viene considerata sbagliata |
| **003** | Ogni risposta corretta assegna un punteggio al giocatore, in base alla difficoltà scelta però, le risposte corrette hanno un valore diverso tra loro: “Difficile” 20 punti, “Medio” 10 punti e “Facile” 5 punti |
| **004** | Se in una domanda ci sono più risposte corrette e l’utente non inserisce il numero giusto di risposte, o le inserisce, ma sbagliate, il punteggio viene calcolato in base al numero di risposte corrette date e al numero di risposte corrette possibili |
| **004** | Dopo aver inserito al meno una risposta, l’utente deve premere “Visualizza Risposta” per verificare la correttezza della risposta |
| **005** | Dopo aver visualizzato la risposta, l’utente deve premere “Procedi” per passare alla domanda successiva |

|  |  |
| --- | --- |
| **ID: REQ-05** | |
| **Nome** | Visualizzazione risultati |
| **Priorità** | 3 |
| **Versione** | 1.0 |
| **Note** | Al termine del quiz, l’utente può vedere la classifica e il proprio punteggio |
| **Sotto requisiti** | |
| **001** | L’utente può visualizzare la classifica dei 10 migliori giocatori e del proprio punteggio |
| **002** | L’utente può premere “Visualizza dettagli”, che offre la possibilità di visualizzare i dettagli delle varie risposte alle domande |
| **003** | L’utente può premere “Nuova Partita”, che permette all’utente di poter continuare a giocare con lo stesso giocatore, ma con la possibilità di cambiare le impostazioni |
| **004** | L’utente può premere “Nuovo Giocatore”, che permette all’utente di creare un nuovo giocatore con cui giocare |
| **004** | L’utente può premere “Chiudi Gioco”, che chiude il gioco |
| **ID: REQ-06** | |
| **Nome** | Visualizzazione dettagli |
| **Priorità** | 4 |
| **Versione** | 1.0 |
| **Note** | L’utente può visualizzare i dettagli di ogni domanda |
| **Sotto requisiti** | |
| **001** | Vengono caricati i dettagli di ogni domanda, con i loro rispettivi punteggi |
| **002** | Dopo aver visualizzato i dettagli, l’utente può premere “Indietro” per tornare alla pagina dei risultati |

**Spiegazione elementi tabella dei requisiti:**

**Nome**: breve descrizione del requisito

**Priorità**: indica l’ordine in cui viene eseguito quel requisito secondo dei numeri

**Versione**: indica la versione del requisito. Ogni modifica apportata avrà una versione aggiornata.

**Note**: descrizione un po’ più dettagliata

**Sotto requisiti**: elementi che compongono il requisito.

## Use case

Figura 1: Use Case

## Pianificazione

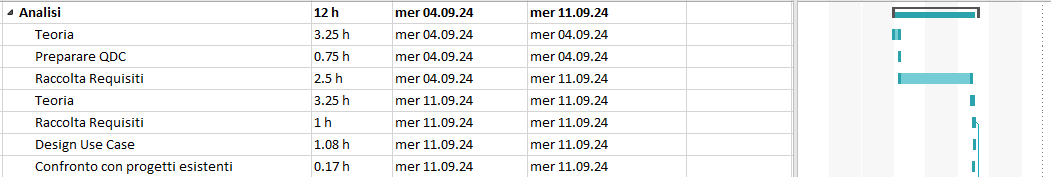
Nell’immagine che segue abbiamo l’analisi, la base per poi iniziare il progetto:

Figura 2: Analisi Gantt Preventivo

Qui sotto invece, troviamo la progettazione del progetto (il predecessore 6 significa che quelle attività dipendono dall’analisi dei requisiti):

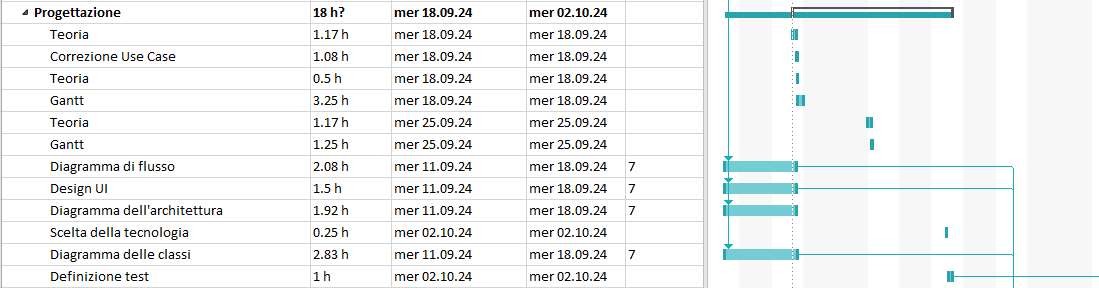


Figura 3: Progettazione Gantt Preventivo

A seguire abbiamo l’implementazione del progetto, cioè la sua creazione (I predecessori 16,17,20 significano che l’implementazione dipende dal Diagramma di flusso, dal Design UI e dal Diagramma delle classi):

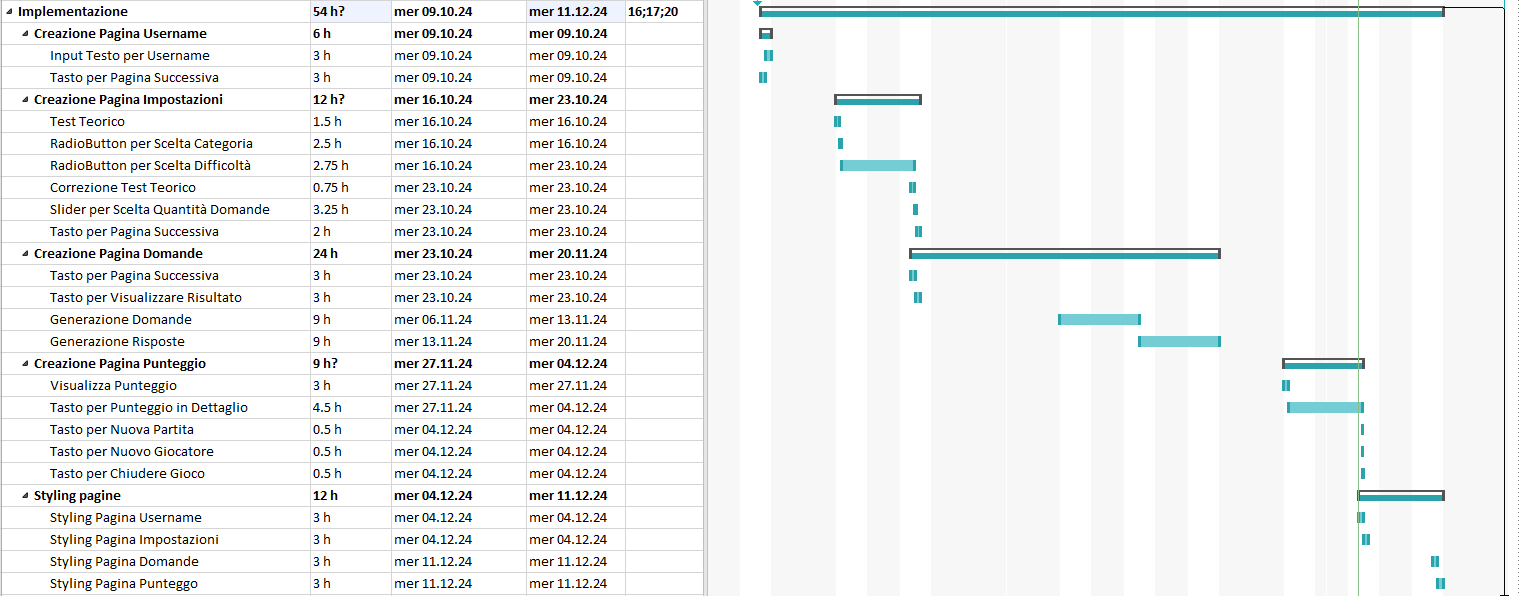


Figura 4: Implementazione Gantt Preventivo

Poi ci sono i Test, per verificare la correttezza del codice:

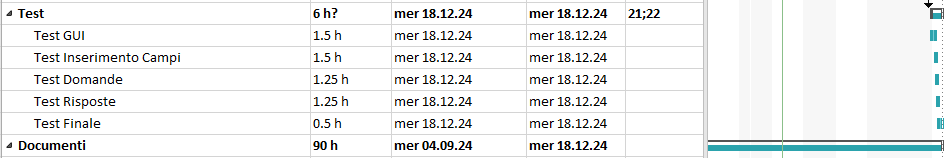


Figura 5: Test Gantt Preventivo

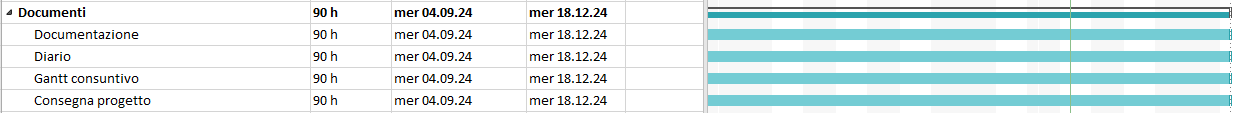
Infine ci sono i Documenti, che verranno compilati durante il progetto:

Figura 6: Documenti Gantt Preventivo

## Analisi dei mezzi

### Software

Il software include:

* JDK: per sviluppare il codice.
* NetBeans 20come Ambiente di Sviluppo Integrato (IDE) per scrivere e gestire il codice.

### Hardware

L’hardware include:

* PC di scuola per la realizzazione:
  + CPU : 13th Gen Intel(R) Core(TM) i7-13700
  + GPU : Intel(R) UHD Graphics 770
  + RAM : 32GB
  + SSD : 512GB
* PC di casa sempre per la realizzazione, per poter continuare in caso di ore saltate a lezione

# Progettazione

## Design dell’architettura del sistema

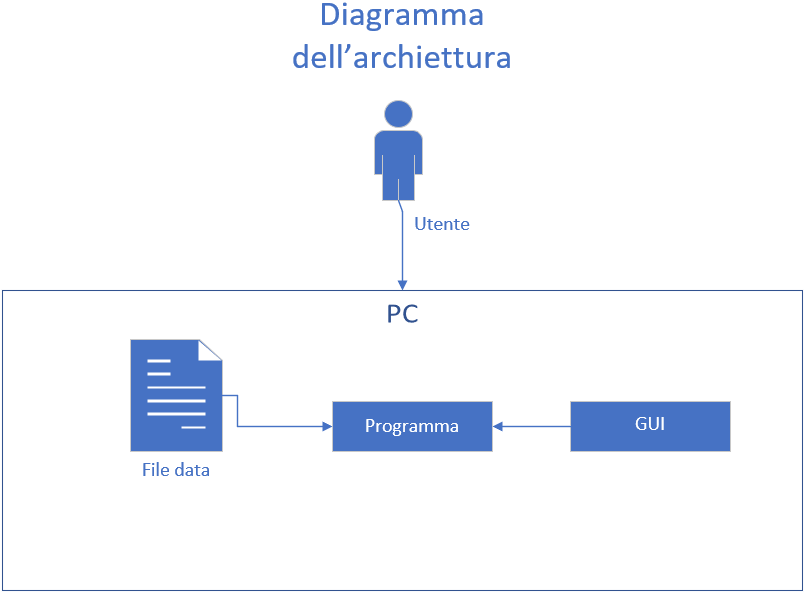


Figura 7: Design dell'architettura

* La struttura del sistema lo schema di rete.
* I componenti che lo compongono.
* I flussi di informazione in ingresso ed in uscita e le relative elaborazioni.

## Design delle interfacce



Figura 8: Parte 1 Pagina Username

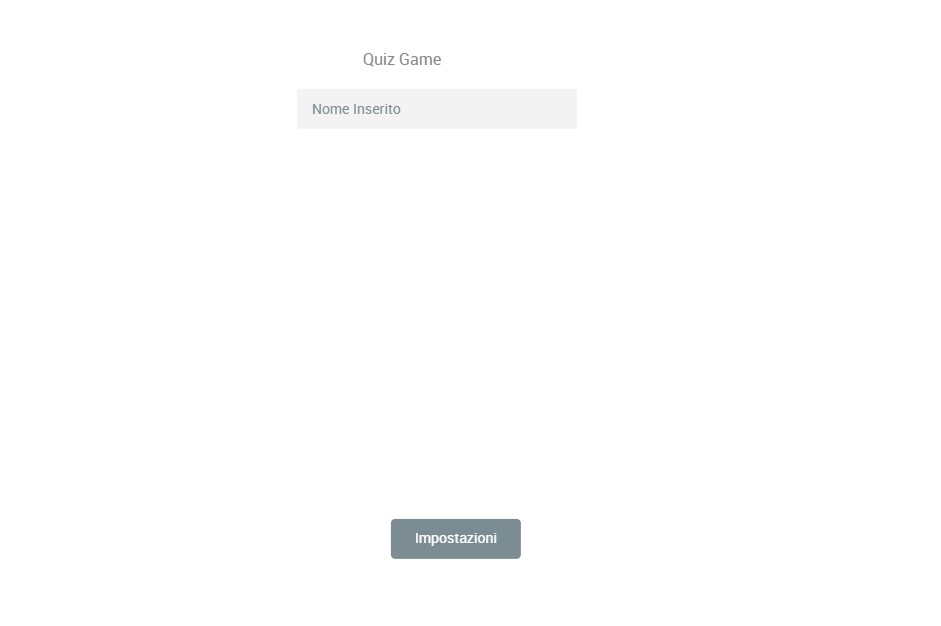


Figura 9: Parte 2 Pagina Username

Queste due Interfacce rappresentano la pagina **“Username”**, dove l’utente, una volta inserito il nome, potrà schiacciare il tasto **“Impostazioni”.**

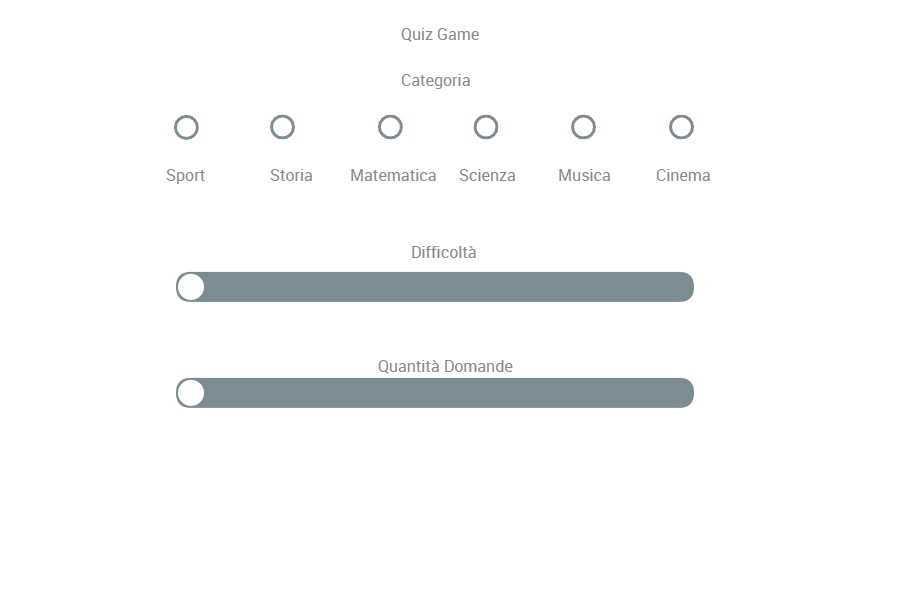


Figura 10: Parte 1 Pagina Impostazioni



Figura 11: Parte 2 Pagina Impostazioni

Le due interfacce successive rappresentano la pagina **“Impostazioni”**, l’utente dovrà inserire dei valori in tutti e 3 i campi per poter premere il tasto e passare finalmente al gioco.

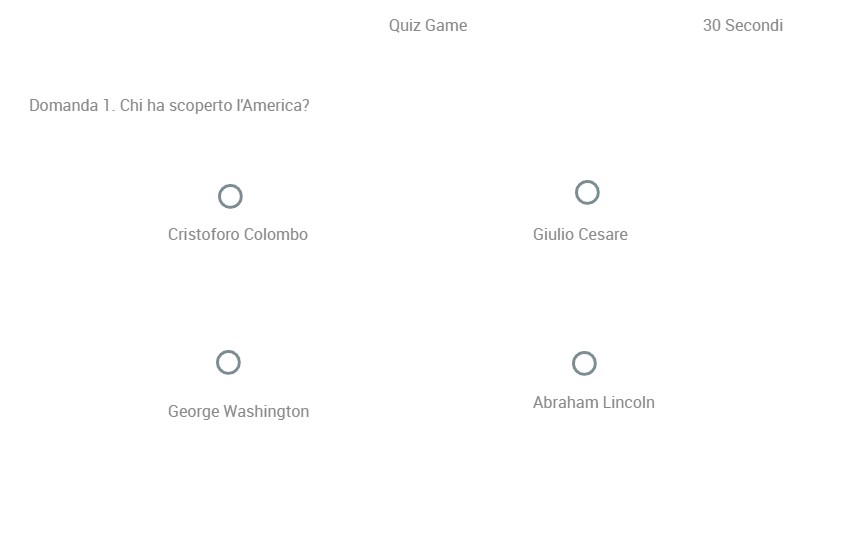


Figura 12: Parte 1 Pagina Domande

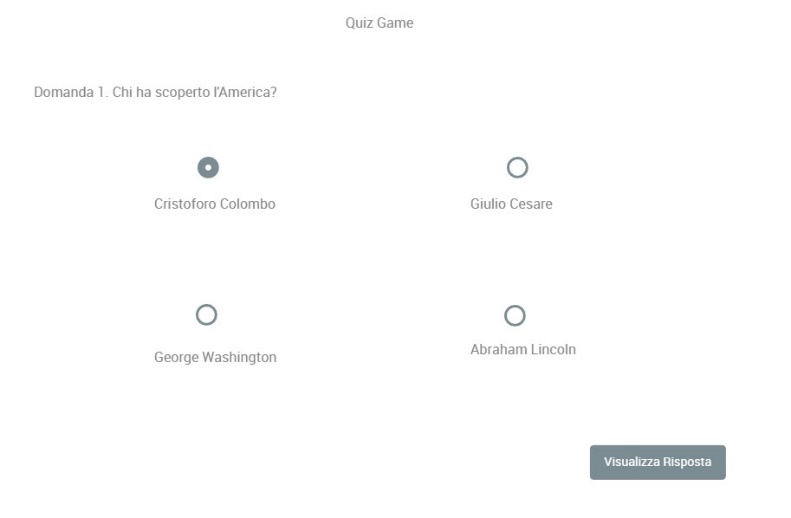


Figura 13: Parte 2 Pagina Domande

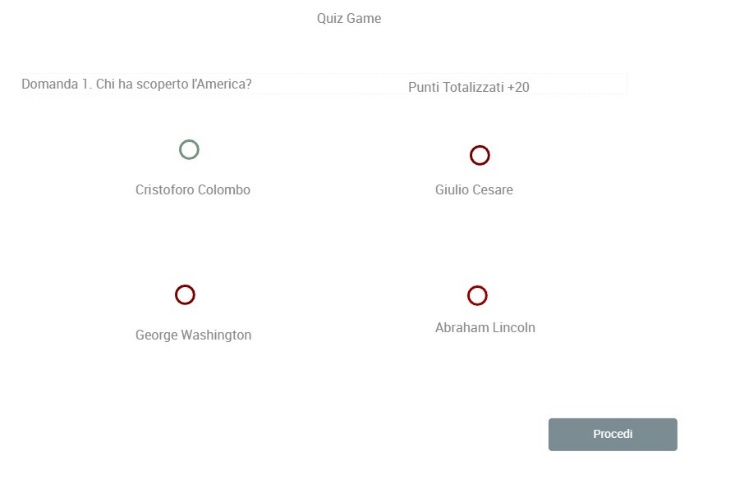


Figura 14: Parte 3 Pagina Domande

Queste tre interfacce rappresentano la pagina **“Domande”**, dove in base alla difficoltà e alla categoria scelta in precedenza ti escono domande diverse, e in base alla quantità di domande scelte, ti escono quel numero di domande. L’utente deve scegliere al meno una risposta per poter visualizzare la risposta corretta e poi procedere alla prossima domanda oppure a visualizzare il punteggio totalizzato.

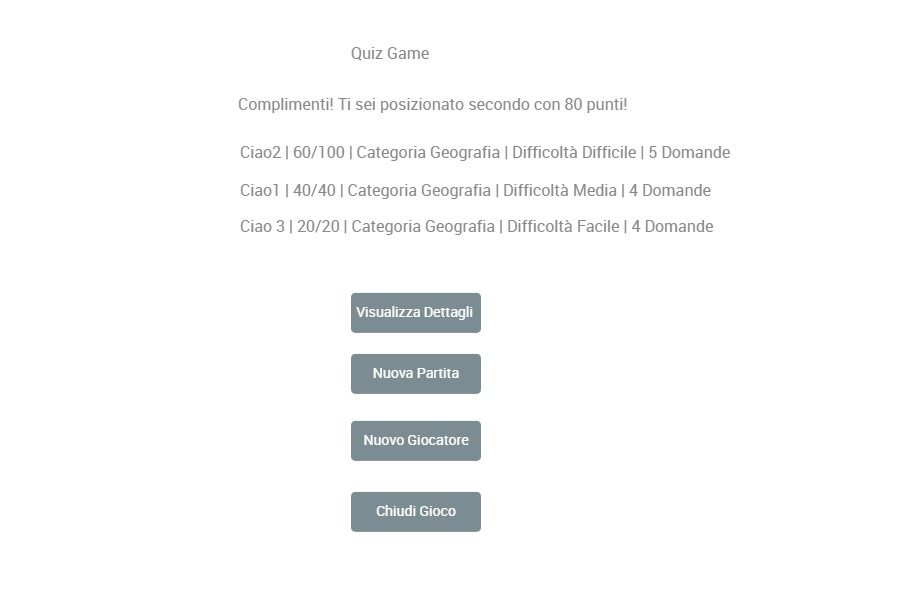


Figura 15: Pagina Risultati

Questa interfaccia rappresenta la pagina **“Risultati”**, l’utente può visualizzare i punteggi totalizzati in quella partita e in che posizione si è posizionato nella classifica, l’utente può comunque decidere di visualizzare i dettagli della partita, di iniziare una nuova partita con lo stesso utente, o di giocarne un’altra ma con un altro giocatore, o di chiudere l’applicazione.

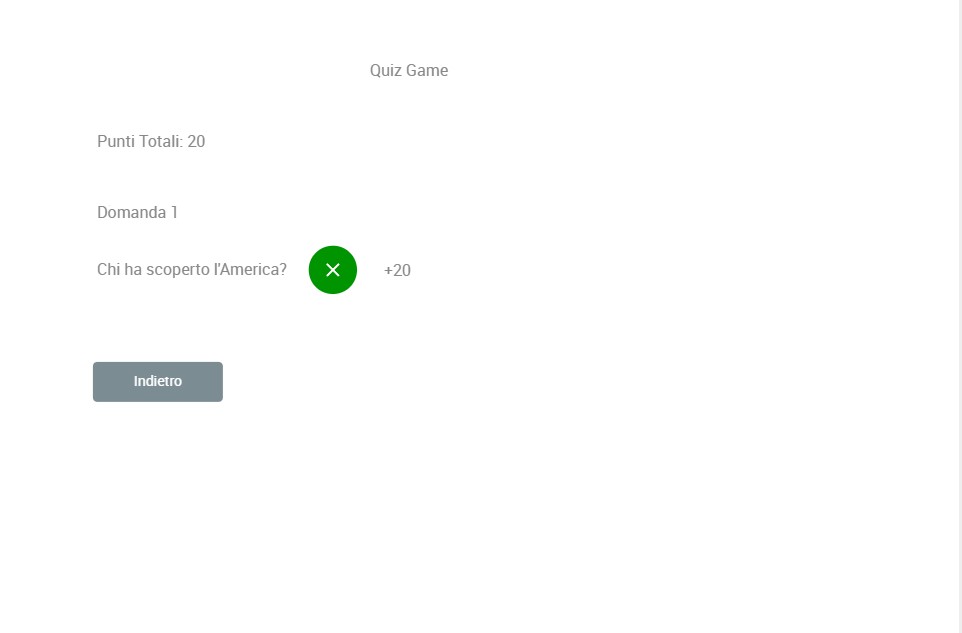


Figura 16: Pagina Dettagli

Quest’ultima interfaccia rappresenta la pagina **“Dettagli”**, l’utente può visualizzare i dettagli di ogni singola domanda della partita appena giocata.

## Design procedurale

Descrive i concetti dettagliati dello sviluppo:

* Diagrammi di flusso
* Diagramma delle classi

### Diagramma di flusso

Figura 17: Diagramma di flusso

### Diagramma delle classi

Figura 18: Diagramma delle classi

# Implementazione

## Classe Main

La classe Main è la classe principale dell’applicativo, dove parte tutto, infatti, serve per poter avviare l’applicazione. Tramite SwingUtilities.inokeLater richiama la schermata UsernamePanel sull’interfaccia grafica. **È stato richiesto l’aiuto di ChatGPT**

public class MainApp {

public static void main(String[] args) {

// Creazione di UsernamePanel sul thread dell'interfaccia grafica, ChatGPT

SwingUtilities.invokeLater(() -> new UsernamePanel());

}

}

## Classe UsernamePanel

La classe UsernamePanel mostrerà la pagina iniziale. L’utente dovrà inserire tramite un JTextField il suo nome, dopo averlo inserito apparirà il tasto che servirà per passare alle impostazioni. Per far apparire il tasto, si usa un KeyListener per rilevare gli input della tastiera e un KeyReleased che controlla quando un tasto viene rilasciato nel JTextField. Per salvare il nome e passare alla schermata successiva, si deve schiacciare sul tasto che richiamerà la funzione per eseguire questi due passaggi.

// funzione per salvare lo username

public void saveUsername() {

String username = usernameInput.getText().trim(); // rimuove spazi iniziali e finali

if (username.isEmpty()) {

username = "Nome Sconosciuto"; // imposta nome di default

}

try {

Path path = Paths.get("src/quiz\_game/username.txt"); // memorizza file in cui salverà il nome

Files.writeString(path, username); // scrive il nome nel file

setVisible(false);

new SettingsPanel(); // apre schermata impostazioni

} catch (IOException ex) {

System.out.println("Errore nel salvataggio del nome");

}

}

## Classe SettingsPanel

La Classe SettingsPanel illustrerà le impostazioni che dovrà configurare l’utente. L’utente dovrà inserire tramite dei jRadioButton la categoria, tramite uno slider sceglierà la difficoltà e con un altro sceglierà la quantità di domande. Per ogni categoria viene creato un JRadioButton che verrà aggiunto al ButtonGroup in modo da poter selezionare solo un’opzione. Per riuscire a mostrare a schermo il valore della difficoltà e della quantità delle domande e per riuscire a portare nel quiz questi dati, viene usato un ChangeListener per rilevare il valore dello slider e uno stateChanged per verificare quando il valore dello slider cambia. Dopo essere apparso il tasto e averlo premuto, verrà richiamata una funzione per poter leggere il valore dal ButtonGroup, dopo aver letto questo valore, verranno salvate le impostazioni nel quiz.

// prende la categoria

public String getCategory() {

ButtonModel selectedModel = categoryGroup.getSelection(); // pulsante selezionato

if (selectedModel != null) {

return selectedModel.getActionCommand(); // ritorna il pulsante selezionato

}

return null;

}

## Classe QuizPanel

La classe QuizPanel mostrerà tutte le domande richieste dall’utente, che verranno filtrate tramite difficoltà e categoria sempre scelta dall’utente. L’utente dovrà inserire tramite dei JRadioButton e dei JCheckBox al meno una risposta, per rendere visibile il tasto che premendo, passerà alla domanda successiva, fino ad arrivare alla schermata dei risultati. Tramite la funzione loadQuestions(), passerà tutte le domande filtrate in una lista. Per mostrare tutte le domande filtrate, si userà la funzione DisplayQuestion(). Infine, per aggiungere i punteggi ed evidenziare le risposte di rosso o di verde si userà la funzione pointAnswer().

### Funzione loadQuestions

La funzione memorizza tutte le righe del file delle domande in unica lista, questa lista viene divisa in un array di stringhe tramite la funzione split(“\\|”), dopo aver ricavato indice per indice nelle variabili, queste variabili vengono salvate in una lista contente tutte le domande.

Domanda tipo in file domande: singola|Matematica|Facile|Quanto fa 2 + 2?|3,4,5,6|1

Questa lista, contenente tutte le domande, viene passata domanda per domanda tramite un foreach, se questa domanda soddisfa la categoria e la difficoltà scelte dall’utente, aggiunge alla lista filtrata la domanda. Dopo aver filtrato tutte le domande, vengono mescolate in modo casuale tramite la funzione Collections.shuffle e poi troncate in base al numero di domande richieste dall’utente tramite la funzione sublist, **trovata tramite ChatGPT**.

// carica in una lista il numero di domande scelte dall'utente in base alla categoria e alla difficoltà

public List<Question> loadQuestions(String category, int difficulty, int quantity) {

List<Question> allQuestions = new ArrayList<>();

Path path = Paths.get("src/quiz\_game/questions.txt");

try {

List<String> lines = Files.readAllLines(path); // legge tutte le righe del file

for (String line : lines) { // passa riga per riga

try {

String[] parts = line.split("\\|"); // divide la riga nei vari campi

String type = parts[0]; // tipo di pulsanti

String questionCategory = parts[1]; // categoria della domanda

String questionDifficulty = parts[2]; // difficoltà della domanda

String text = parts[3]; // testo della domanda

List<String> options = Arrays.asList(parts[4].split(",")); // divide il campo nelle opzioni possibili e li passa sottoforma di lista

List<Integer> correctAnswers = new ArrayList<>();

// essendo che serve una lista di numeri e le risposte sono sottoforma di String,

// passa risposta per risposta e le trasforma in Integer per poi passarle alla lista

for (String index : parts[5].split(",")) {

correctAnswers.add(Integer.parseInt(index));

}

// aggiunge la domanda alla lista

allQuestions.add(new Question(type, questionCategory, questionDifficulty, text, options, correctAnswers));

} catch (Exception e) {

System.out.println("Errore durante il caricamento di una domanda: " + line);

}

}

// aggiunge le domande filtrate a seconda della categoria e della difficoltà che richiede l'utente

List<Question> filteredQuestions = new ArrayList<>();

for (Question question : allQuestions) {

// aggiunge alla lista filtrata le domande che comprendono quel livello di difficoltà e quella categoria

if (question.getCategory().equals(category) && convertDifficulty(question.getDifficulty()) == difficulty) {

filteredQuestions.add(question);

}

}

// mescola le domande

Collections.shuffle(filteredQuestions);

// ritorna il numero di domande, sempre sottoforma di lista, che richiedeva l'utente, ChatGPT

return filteredQuestions.subList(0, Math.min(quantity, filteredQuestions.size()));

} catch (IOException e) {

System.out.println("Errore durante il caricamento del file delle domande");

return Collections.emptyList();

}

}

### Funzione displayQuestions

La funzione displayQuestions() controlla come prima cosa, se non siano finite le domande in modo da passare alla schermata dei risultati una volta finite passando il punteggio datto dall’utente, le domande filtrate, le risposte date dall’utente e le risposte corrette. Viene passata la domanda corrente nella variabile question di tipo Question (Classe domanda). Mostra il testo della domanda, e in base al tipo della domanda e a quante opzioni ci sono, mostra quel numero di opzioni formato di soli JRadioButton o di soli JCheckBox. Viene aggiunto a tutti i JRadioButton o i JCheckBox il listener tramite un ciclo che passa pulsante per pulsante, per i JRadioButton appartenenti al ButtonGroup, vengono salvati in una lista per riuscire a ciclarli. Grazie a questi listener aggiunti ai pulsanti, una volta cliccato anche solo un pulsante farà partire il listener che renderà visibile il tasto, per i JRadioButton controllava se uno degli elementi del ButtonGroup era stato selezionato, per i JCheckBox, **dove ho usato ChatGPT**, si applica la funzione stream() per permettere l’iterazione dei dati, poi la funzione anyMatch() per verificare se la condizione all’interno fosse vera e la condizione AbstractButton::isSelected controllava se al meno uno dei pulsanti fosse stato selezionato. Dopo aver premuto sul tasto passa alla funzione pointAnswer(). A ogni chiamata di questa funzione, revalidate() aggiornerà il layout e repaint() ridisegnerà la finestra, **trovato tramite ChatGPT.**

// mostra le domande salvate nella lista precedente

public void displayQuestion(int diff) {

// se le domande sono finite

if (currentQuestion >= questions.size()) {

new ResultPanel(score, questions, userAnswers, questionScores); // apre schermata quiz passando punteggio,

// lista di domande, lista di risposte e lista di punteggi

setVisible(false);

return; // esce dalla funzione

}

// ottiene domanda corrente

Question question = questions.get(currentQuestion);

// rimuove tutti i componenti precedenti dalla finestra

getContentPane().removeAll();

GridBagConstraints gbc = new GridBagConstraints();

gbc.insets = new Insets(10, 10, 10, 10);

gbc.fill = GridBagConstraints.HORIZONTAL;

// mostra numero della domanda corrente e testo della domanda

JLabel questionLabel = new JLabel((currentQuestion + 1) + ". " + question.getText());

questionLabel.setFont(new Font("SansSerif", Font.BOLD, 16));

gbc.gridx = 0;

gbc.gridy = 0;

gbc.gridwidth = 2;

add(questionLabel, gbc);

JPanel optionsPanel = new JPanel();

optionsPanel.setLayout(new GridLayout(0, 1));

ButtonGroup buttonGroup = new ButtonGroup();

List<JCheckBox> checkboxes = new ArrayList<>(); // lista per checkbox,

// non ho fatto la lista per i radio button perché per i radiobutton

// puoi mettere solo una risposta

// mappa per salvare tasto e indice del tasto

Map<AbstractButton, Integer> buttonIndices = new HashMap<>();

// mostra tutte le opzioni

for (int i = 0; i < question.getOptions().size(); i++) {

String option = question.getOptions().get(i); // prende l'opzione in quell'indice

if (question.getType().equals("singola")) { // decide se mettere radiobutton o checkbox

JRadioButton radioButton = new JRadioButton(option); // rende l'opzione un radiobutton

radioButton.setActionCommand(String.valueOf(i)); // imposta al bottone posizione i,

//il valore della sua apposita opzione

buttonGroup.add(radioButton);

optionsPanel.add(radioButton);

buttonIndices.put(radioButton, i);

} else {

JCheckBox checkBox = new JCheckBox(option); // rende l'opzione un checkbox

checkboxes.add(checkBox);

optionsPanel.add(checkBox);

buttonIndices.put(checkBox, i);

}

}

gbc.gridy = 2;

add(optionsPanel, gbc);

JButton answerButton = new JButton("Visualizza Risposta");

answerButton.setFont(new Font("SansSerif", Font.BOLD, 14));

answerButton.setBackground(new Color(0, 0, 0));

answerButton.setForeground(new Color(245, 245, 255));

answerButton.setVisible(false);

// listener per rendere visibile il tasto solo quando una risposta è selezionata

ActionListener visibleButton = e -> {

if (question.getType().equals("singola")) {

answerButton.setVisible(buttonGroup.getSelection() != null);

}

// ChatGPT

else {

// stream permette l'iterazione sui dati

// anyMatch controlla se un al meno soddisfa la condizione

// AbstractButton::isSelected metodo che fa il controllo su ogni elemento se è selezionato

boolean anySelected = checkboxes.stream().anyMatch(AbstractButton::isSelected);

answerButton.setVisible(anySelected); // rende visibile il tasto se anySelected è true,

// quindi se uno dei checkbox è selezionato

}

};

// aggiunge il listener ai tasti delle opzioni

if (question.getType().equals("singola")) {

// inserisce tutti gli elementi appartenenti al gruppo di tasti in una lista

List<AbstractButton> buttonsList = Collections.list(buttonGroup.getElements());

for (AbstractButton button : buttonsList) {

// aggiunge il listener ai tasti

button.addActionListener(visibleButton);

}

} else {

for (JCheckBox checkBox : checkboxes) {

checkBox.addActionListener(visibleButton);

}

}

// listener per il pulsante "Visualizza Risposta"

answerButton.addActionListener(e -> {

pointAnswer(answerButton, buttonIndices, question.getCorrectAnswers(), diff);

});

gbc.gridy = 3;

add(answerButton, gbc);

revalidate(); // aggiorna il layout, ChatGPT

repaint(); // ridisegna la finestra, ChatGPT

}

### Funzione pointAnswer

La funzione controlla come prima cosa che le domande non siano finite. Viene iterato chiave e valore di tutti i bottoni. Ogni chiave viene passata in btn di tipo AbstractButton, si controlla se btn è selezionato e se è stato selezionato, lo aggiungerà alla lista delle risposte scelte dall’utente. Ogni valore viene passa in index di tipo int, si controlla se la lista di risposte corrette contenga il pulsante, e se lo contiene lo colora di verde, altrimenti di rosso. Per il punteggio, viene calcolato a seconda della difficoltà e dalle possibili risposte corrette e quante realmente ne ha selezionate correttamente l’utente. Dopo aver premuto sul tasto, il listener ritornerà sulla funzione displayQuestion() ma con la domanda successiva.

private void pointAnswer(JButton button, Map<AbstractButton, Integer> buttonIndices, List<Integer> correctAnswers, int diff) {

if (currentQuestion >= questionScores.size()) {

System.out.print("");

return;

}

List<Integer> selectedAnswers = new ArrayList<>();

int correctCount = 0;

// buttonIndices passa chiave e valore a entry

for (Map.Entry<AbstractButton, Integer> entry : buttonIndices.entrySet()) {

// btn prende la chiave di entry

AbstractButton btn = entry.getKey();

// index prende il valore di entry (indice bottone)

int index = entry.getValue();

if (btn.isSelected()) {

// se selezionato, aggiunge alla lista dei bottoni selezionati index

selectedAnswers.add(index);

}

if (correctAnswers.contains(index)) {

// se il bottone è una delle risposte corrette,

// colora di verde il testo, sennò lo colora di rosso

btn.setForeground(Color.GREEN);

if (btn.isSelected()) {

correctCount++;

}

} else if (btn.isSelected()) {

btn.setForeground(Color.RED);

}

}

int totalCorrect = correctAnswers.size();

int questionScore = 0;

if (diff == 1) {

questionScore = 10 \* correctCount;

}

else if (diff == 2) {

if (totalCorrect == 1) {

questionScore = correctCount \* 7;

}

else if (totalCorrect == 2) {

questionScore = correctCount \* 10;

}

else {

questionScore = correctCount \* 7;

if (correctCount == 3) {

questionScore = 20;

}

}

}

else if (diff == 3) {

if (totalCorrect == 1) {

questionScore = correctCount \* 10;

}

else if (totalCorrect == 2) {

questionScore = correctCount \* 15;

}

else {

questionScore = correctCount \* 10;

}

}

score += questionScore;

// inserisce quel punteggio a quella posizione

questionScores.set(currentQuestion, questionScore);

// inserisce quella risposta a quella posizione

userAnswers.set(currentQuestion, selectedAnswers);

button.setText("Procedi");

button.addActionListener(e -> {

currentQuestion++; // incrementa l'indice, per poter prende la domanda successiva

displayQuestion(diff); // richiama la funzione per poter mostrare la domanda

});

}

## Classe ResultPanel

La classe ResultPanel mostrerà la classifica dei primi 10 classificati al quiz. All’interno della classe prendiamo la classifica aggiornata col nuovo utente tramite la classe LeaderboardManager(), dove passiamo il nome, che si prenderà tramite la funzione getPlayerName(), e il punteggio. Dopo aver aggiornato la classifica, iteriamo giocatore per giocatore, tramite un foreach, e lo si mostra a schermo. Si aggiunge alla classifica, la capacità di poter scrollare la classifica tramite un JScrollPane, **preso da ChatGPT**. Si aggiungono 4 tasti per alla schermata, uno per visualizzare i dettagli dove richiama la classe DetailsPanel(), uno per iniziare una nuova partita dove richiama la classe SettingsPanel(), uno per creare un nuovo giocatore dove richiama la classe UsernamePanel() e uno per chiudere l’applicativo.

## Classe LeaderboardManager

La classe LeaderboardManager() gestisce la classifica. Una volta richiamata la funzione updateLeaderboard(), crea una lista dei giocatori già esistenti richiamando la funzione loadLeaderboard con lo stesso procedimento già visto nella funzione loadQuestions(). Aggiunge alla lista il nuovo utente mette in ordine decrescente la classifica, dal migliore al peggiore, **Fatto con ChatGPT.** Tronca la classifica per fare in modo che nella classifica si visualizzino i 10 migliori giocatori. Infine salva la classifica aggiornata nel file richiamando la funzione saveLeaderboard(), dove converte la lista dei 10 migliori giocatori in una lista di stringhe formattate e per ogni elemento della lista si estrae il nome e il punteggio.

## Classe DetailsPanel

La Classe DetailsPanel mostra in dettaglio ogni domanda che ha fatto il giocatore. Dalla lista delle domande che è stata passata dalla classe ResultPnale(), si ricava domanda per domanda, dalla domanda si ricava la risposta data dall’utente e quella corretta tramite la funzione formatAnswer(), e il punteggio totalizzato. Premendo il tasto “Indietro” torna alla schermata ResultPanel();

# Test

## Protocollo di test

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case**  **Riferimento** | TC-001  REQ-01 | **Nome** | Inserimento nome giocatore |
| **Descrizione** | Apparizione tasto “Impostazioni” e salvataggio in file | | |
| **Prerequisiti** | File salvataggio già creato | | |
| **Procedura** | * Inserire un nome tramite input text * Verificare che il campo venga salvato nel file di tipo txt * Verificare che una volta che il nome venga inserito, il tasto “Impostazioni” appaia | | |
| **Risultati attesi** | Se viene inserito un nome, il tasto “Impostazioni” appare e una volta premuto, il nome viene salvato nel file di tipo txt | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case**  **Riferimento** | TC-002  REQ-02 | **Nome** | Inserimento Impostazioni |
| **Descrizione** | Apparizione tasto “Inzia Gioco” | | |
| **Prerequisiti** |  | | |
| **Procedura** | * Selezionare la categoria, la difficoltà e la quantità delle domande * Verificare che il tasto “Inizia Gioco” appaia, dopo aver inserito tutte le impostazioni | | |
| **Risultati attesi** | Il tasto “Inizia Gioco” deve apparire solo quando inserisco tutte le impostazioni | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case**  **Riferimento** | TC-003  REQ-03 | **Nome** | Inizio del gioco |
| **Descrizione** | L’utente deve rispondere alle domande che li vengono proposte | | |
| **Prerequisiti** |  | | |
| **Procedura** | * Verificare che le domande siano caricate correttamente in base alla categoria e alla difficoltà scelta * Verificare che le risposte siano correlate in base alla domanda e che le risposte sbagliate vengono prese in modo casuale * Verificare la presenza di immagini in alcune domande e risposte | | |
| **Risultati attesi** | Le domande vengono caricate correttamente, le risposte sono prese dalla domanda giusta e le risposte sbagliate sono prese in modo casuale, e la presenza delle immagini | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case**  **Riferimento** | TC-004  REQ-04 | **Nome** | Inserimento risposte |
| **Descrizione** | L’utente deve selezionare le risposte tramite checkbox o radiobutton | | |
| **Prerequisiti** |  | | |
| **Procedura** | * Verificare che le domande con più risposte siano dei checkbox e quelle con una risposta siano un radiobutton * Verificare che una volta finito il tempo per rispondere a una domanda, la risposta viene considerata automaticamente sbagliata * Verificare la giusta distribuzione di punti in base alla difficoltà scelta * Verificare che, se non si ha risposto a tutte le risposte corrette di quella domanda, il punteggio calcolato in base al numero di risposte corrette date e al numero di risposte corrette possibili * Verificare che all’inserimento di una sola risposta, appaia il tasto “Visualizza Risposta” * Verificare che, dopo aver premuto “Visualizza Risposta”, il tasto cambi in “Procedi” | | |
| **Risultati attesi** | Checkbox appare quando ci sono più risposte possibili e radiobutton quando c’è una sola risposta. La risposta venga considerata sbagliata una volta finito il tempo. I punti vengono distribuiti in modo corretto. I punti vengono calcolati in base al numero di risposte corrette date e al numero di risposte corrette possibili. Il tasto “Visualizza Risposta” appare quando inserisco al meno una risposta. Il tasto “Procedi” cambia quando si preme su “Visualizza Risposta” | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case**  **Riferimento** | TC-005  REQ-05 | **Nome** | Visualizzazione Risultati |
| **Descrizione** | Al termine del quiz, l’utente può vedere la classifica e il proprio punteggio | | |
| **Prerequisiti** |  | | |
| **Procedura** | * Visualizzare la classifica dei 10 migliori * Verificare che il tasto “Visualizza Dettagli” ti faccia vedere i dettagli delle varie domande * Verificare che il tasto “Nuova Partita” ti faccia cambiare le impostazioni del gioco, ma con le stesse domande * Verificare che il tasto “Nuovo Giocatore” ti faccia inserire un nuovo giocatore * Verificare che il tasto “Chiudi Gioco” ti faccia chiudere il gioco | | |
| **Risultati attesi** | Il tasto “Visualizza Dettagli” fa vedere i dettagli delle domande. Il tasto “Nuova Partita” fa inserire delle nuove impostazioni, ma con lo stesso giocatore. Il tasto “Nuovo Giocatore” fa creare un nuovo giocatore. Il tasto “Chiudi Gioco” chiude il gioco | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case**  **Riferimento** | TC-006  REQ-06 | **Nome** | Visualizzazione Dettagli |
| **Descrizione** | L’utente può visualizzare i dettagli di ogni domanda | | |
| **Prerequisiti** |  | | |
| **Procedura** | * Visualizzare dettagli delle domande * Verificare che il tasto “Indietro”, torni alla pagina dei risultati | | |
| **Risultati attesi** | Il tasto “Indietro” torna alla pagina dei risultati | | |

## Risultati test

|  |  |  |
| --- | --- | --- |
| **Test Case** | **Risultato ottenuto** | **Stato** |
| TC-001 | Appare tasto quando viene inserito il nome e salva il nome nel file | Passato |
| TC-002 | Il tasto non appare, è già presente | Parzialmente Passato |
| TC-003 | Domande e risposte vengono caricate correttamente in base alla categoria e alla difficoltà scelta. La cosa non funzionante sono le immagini all’interno delle domande e delle risposte | Parzialmente Passato |
| TC-004 | Domande con più possibili risposte sono checkbox e domande con una singola risposta sono radiobutton. Viene distribuito la giusta quantità di punti in base alla difficoltà e al numero di risposte date. Appare tasto dopo aver inserito al meno una risposta. Cambia testo del tasto dopo aver premuto su di esso. Il timer non viene implementato durante le domande | Parzialmente Passato |
| TC-005 | Visualizza la classifica coi 10 migliori giocatori e i tasti eseguono tutti le loro funzionalità | Passato |
| TC-006 | Visualizza tutti i dettagli di tutte le domande e il tasto torna alla pagina dei risultati | Passato |

## Mancanze/limitazioni conosciute

Gli elementi mancanti sono le immagini per domande e risposte e il timer per ogni risposta, non sono riuscito a implementarli per troppe complicazioni e anche per tempo un po’ ristrettivo

# Consuntivo

## Progettazione

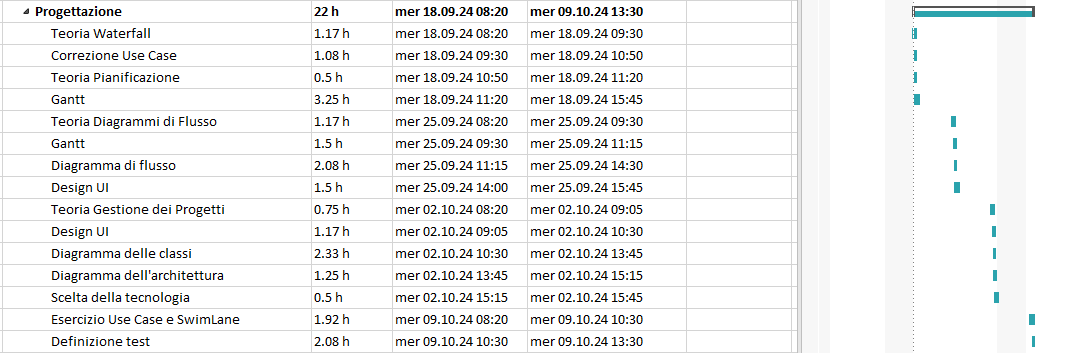


Figura 19: Analisi Gantt Preventivo

La Progettazione ha avuto una durata molto simile a quella del Gantt Preventivo, è durato leggermente di più perché ho lavaorato maggiormente al Design UI e alla Definizione dei Test.

## Implementazione

Figura 20: Implementazione 1 Gantt Preventivo



Figura 21: Implementazione 2 Gantt Preventivo

L’Implementazione è leggermente minore rispetto al Gantt Preventivo, in quanto ho lavorato alla documentazione più di quanto mi aspettassi

## Test



Figura 22: Test Gantt Preventivo

I Test sono durati molto di meno rispetto al Gantt Preventivo, perché durante l’esecuzione del codice lavoravo già molto per vedere le funzionalità che non andavano e arrivato il momento di documentarlo sapevo già testare in modo rapido. Questo tempo in più l’ho dedicato alla documentazione.

# Conclusioni

Questo progetto è stato realizzato per le persone che hanno piacere di giocare a un quiz su vari argomenti. L’uso dell’applicativo è molto veloce e semplice da capire. Può essere utile nell’aiuto della creazione dei prossimi progetti. Inoltre, questo progetto ha un successo importante nel percorso di tutti noi, perché possiamo dire di avere realizzato il nostro primo progetto in assoluto.

## Sviluppi futuri

Come sviluppi futuri, potrei aggiungere all’applicativo gli elementi mancanti o addirittura crearne uno nuovo con all’interno più funzionalità

## Considerazioni personali

Sono parzialmente soddisfatto del risultato del progetto, dato che sono riuscito a soddisfare gran parte dei requisiti.

Questo progetto mi ha aiutato a capire come organizzarmi in base al tempo che si ha a disposizione

Per concludere, questo progetto mi ha aiutato molto nella formazione da informatico, perché ho appreso, per la prima volta, una parte del modo in cui si lavora in questo ambito

# Bibliografia

## Sitografia

1. <https://www.w3schools.com/>
2. <https://www.javatpoint.com/java-swing>
3. <https://www.geeksforgeeks.org/introduction-to-java-swing/>

# Glossario

|  |  |
| --- | --- |
| **Termine** | **Significato** |
| IDE | Integrated Development Environment, software progettato per la realizzazione di applicazioni che aggrega strumenti di sviluppo comuni in un’unica interfaccia utente grafica |
| Java | Linguaggio di programmazione multipiattaforma |
| JDK | Java Development Kit, insieme di software necessari per compilare, eseguire e testare applicazioni |
| Netbeans | Ambiente di sviluppo integrato multi-linguaggio |

# Indice delle figure

[Figura 1: Use Case 10](file:///D:\3°anno\Professionale\M306\Quiz_Game\3_Documentazione\Documentazione_Progetto.docx#_Toc185425592)

[Figura 2: Analisi Gantt Preventivo 11](file:///D:\3°anno\Professionale\M306\Quiz_Game\3_Documentazione\Documentazione_Progetto.docx#_Toc185425593)

[Figura 3: Progettazione Gantt Preventivo 11](#_Toc185425594)

[Figura 4: Implementazione Gantt Preventivo 11](file:///D:\3°anno\Professionale\M306\Quiz_Game\3_Documentazione\Documentazione_Progetto.docx#_Toc185425595)

[Figura 5: Test Gantt Preventivo 12](file:///D:\3°anno\Professionale\M306\Quiz_Game\3_Documentazione\Documentazione_Progetto.docx#_Toc185425596)

[Figura 6: Documenti Gantt Preventivo 12](file:///D:\3°anno\Professionale\M306\Quiz_Game\3_Documentazione\Documentazione_Progetto.docx#_Toc185425597)

[Figura 7: Design dell'architettura 13](#_Toc185425598)

[Figura 8: Parte 1 Pagina Username 15](file:///D:\3°anno\Professionale\M306\Quiz_Game\3_Documentazione\Documentazione_Progetto.docx#_Toc185425599)

[Figura 9: Parte 2 Pagina Username 15](file:///D:\3°anno\Professionale\M306\Quiz_Game\3_Documentazione\Documentazione_Progetto.docx#_Toc185425600)

[Figura 10: Parte 1 Pagina Impostazioni 16](file:///D:\3°anno\Professionale\M306\Quiz_Game\3_Documentazione\Documentazione_Progetto.docx#_Toc185425601)

[Figura 11: Parte 2 Pagina Impostazioni 16](file:///D:\3°anno\Professionale\M306\Quiz_Game\3_Documentazione\Documentazione_Progetto.docx#_Toc185425602)

[Figura 12: Parte 1 Pagina Domande 17](file:///D:\3°anno\Professionale\M306\Quiz_Game\3_Documentazione\Documentazione_Progetto.docx#_Toc185425603)

[Figura 13: Parte 2 Pagina Domande 17](file:///D:\3°anno\Professionale\M306\Quiz_Game\3_Documentazione\Documentazione_Progetto.docx#_Toc185425604)

[Figura 14: Parte 3 Pagina Domande 17](file:///D:\3°anno\Professionale\M306\Quiz_Game\3_Documentazione\Documentazione_Progetto.docx#_Toc185425605)

[Figura 15: Pagina Risultati 18](file:///D:\3°anno\Professionale\M306\Quiz_Game\3_Documentazione\Documentazione_Progetto.docx#_Toc185425606)

[Figura 16: Pagina Dettagli 18](file:///D:\3°anno\Professionale\M306\Quiz_Game\3_Documentazione\Documentazione_Progetto.docx#_Toc185425607)

[Figura 17: Diagramma di flusso 19](file:///D:\3°anno\Professionale\M306\Quiz_Game\3_Documentazione\Documentazione_Progetto.docx#_Toc185425608)

[Figura 18: Diagramma delle classi 20](#_Toc185425609)

[Figura 19: Analisi Gantt Preventivo 35](#_Toc185425610)

[Figura 20: Implementazione 1 Gantt Preventivo 36](file:///D:\3°anno\Professionale\M306\Quiz_Game\3_Documentazione\Documentazione_Progetto.docx#_Toc185425611)

[Figura 21: Implementazione 2 Gantt Preventivo 37](#_Toc185425612)

[Figura 22: Test Gantt Preventivo 38](#_Toc185425613)

# Allegati

Elenco degli allegati:

* QdC
* Abstract
* Diari
* Applicativo
* Diagramma dell’archiettura
* Diagramma delle classi
* Diagramma di flusso
* Gantt
* Gantt Consuntivo
* GUI
* Link
* Scelta della tecnologia
* Use Case