Documentazione del Progetto

Autore: *Inserisci il tuo nome*

11 marzo 2025

Indice

1 Analisi dei Requisiti

1.1 Requisiti Funzionali

1.1.1 Autenticazione e Gestione Utenti

- **Login:** L'utente deve poter accedere inserendo username e password.
- **Registrazione:** Possibilità di creare un nuovo account, con validazione dei dati (username univoco, password sicura, ecc.).
- **Persistenza Utenti:** I dati degli utenti devono essere salvati (es. su file in formato JSON/XML/CSV).

1.1.2 Gestione Esercizi

- **Visualizzazione Esercizi:** Mostrare una dashboard con una griglia di esercizi raggruppati per tipologia (es. "Leggi il Codice", "Trova l'Errore", "Cosa Stampa?").
- **Dettaglio Esercizio:** Visualizzare la descrizione, le regole, il livello attuale e la motivazione dell'esercizio selezionato.
- **Esecuzione Esercizio:** L'utente interagisce con l'esercizio rispondendo a domande o completando attività; deve essere gestita la progressione tra livelli di difficoltà.
- **Salvataggio Risultati:** Alla fine (o in caso di interruzione) salvare i risultati (punteggio, tempo, numero di tentativi, livello raggiunto).

1.1.3 Feedback e Notifiche

- **Feedback Immediato:** Durante l'esecuzione, fornire feedback sulle risposte.
- **Report Finale:** Mostrare un riepilogo dei risultati e suggerimenti per il miglioramento.
- **Notifiche (Opzionale):** Invio di notifiche via email o Telegram riguardo ai progressi o ai risultati.

1.1.4 Funzionalità Extra

- **Modalità Amministratore:** Possibilità di aggiungere, modificare o eliminare esercizi.
- **Leaderboards:** Visualizzazione di una classifica con i migliori punteggi.
- **Multi-Lingua:** Supporto per cambiare la lingua dell'interfaccia.
- **Timer e Backup:** Implementazione di un timer per ogni esercizio e salvataggio periodico del progresso.

1.2 Requisiti Non Funzionali

- **Usabilità:** Interfaccia utente semplice, intuitiva e responsive.
- **Performance:** Tempi di risposta rapidi, soprattutto nelle operazioni di caricamento e salvataggio.
- **Sicurezza:** Gestione sicura delle password (crittografia), validazione dei dati in input e protezione dei file di persistenza.
- **Manutenibilità:** Struttura del codice modulare e ben documentata, utilizzando un'architettura come MVVM per separare logica e interfaccia.
- **Portabilità:** L'applicazione deve essere stand-alone, utilizzabile su diversi sistemi in cui sia presente una JVM.

2 Use Case

2.1 Use Case 1: Autenticazione e Registrazione

Attore Principale: Utente
Descrizione: L'utente può accedere all'app attraverso il login oppure registrarsi come nuovo utente.
Flusso Principale:

1. L'utente apre l'app e viene visualizzata la schermata di login.
2. Inserisce username e password e clicca "Accedi".
3. Il sistema verifica le credenziali e, in caso di successo, redirige alla dashboard.

Flusso Alternativo:

a. L'utente clicca sul pulsante "Crea Nuovo Utente".
b. Compila il modulo di registrazione con username, password, nome, cognome e altri eventuali dati (es. recapito per notifiche).
c. Il sistema valida i dati, li salva e conferma la registrazione, permettendo il successivo login.

2.2 Use Case 2: Accesso alla Dashboard

Attore Principale: Utente autenticato
Descrizione: Dopo il login, l'utente accede a una dashboard che mostra la griglia degli esercizi, il progresso raggiunto e altre funzionalità di navigazione.
Flusso Principale:

1. Il sistema carica i dati dell'utente (esercizi completati, punteggi, livello di avanzamento).
2. Viene presentata la dashboard con icone degli esercizi, barre di avanzamento e link a funzioni extra (es. impostazioni, leaderboard).

2.3 Use Case 3: Selezione e Visualizzazione del Dettaglio Esercizio

Attore Principale: Utente autenticato

Descrizione: L'utente seleziona un esercizio dalla dashboard per visualizzarne il dettaglio.

Flusso Principale:

1. L'utente clicca su un esercizio specifico.
2. Il sistema mostra una schermata di dettaglio con:
    - Descrizione dell'esercizio.
    - Regole e modalità di esecuzione.
    - Livello attuale e criteri di progressione.
    - Pulsanti per "Inizia" o "Annulla".

2.4 Use Case 4: Esecuzione dell'Esercizio

Attore Principale: Utente

Descrizione: L'utente esegue l'esercizio, risponde alle domande o completa le attività previste.

Flusso Principale:

1. Dopo aver cliccato "Inizia", l'esercizio viene caricato.
2. L'utente interagisce con l'interfaccia (inserimento risposte, selezione opzioni, drag-and-drop, ecc.).
3. Il sistema fornisce feedback immediato su ogni risposta e aggiorna la barra di avanzamento.
4. Se l'utente completa il numero richiesto di esercizi, il sistema passa al livello successivo.

Flusso Alternativo:

a. L'utente clicca "Esci dall'esercizio" prima del completamento.

b. Il sistema interrompe l'esercizio, registra il tentativo come fallito e salva il risultato parziale.

2.5 Use Case 5: Visualizzazione dei Risultati e Feedback

Attore Principale: Utente

Descrizione: Al termine dell'esercizio, l'utente riceve un riepilogo dei risultati.

Flusso Principale:

1. Il sistema mostra una schermata di riepilogo con:
    - Punteggio ottenuto.

- Numero di risposte corrette ed errate.
- Tempo impiegato.
- Feedback personalizzato e suggerimenti per il miglioramento.

2. I risultati vengono salvati per future consultazioni.

2.6 Use Case 6: Logout e Chiusura dell'Applicazione

Attore Principale: Utente autenticato

Descrizione: L'utente termina la sessione e il sistema effettua il logout.

Flusso Principale:

1. L'utente clicca sul pulsante "Logout".
2. Il sistema salva ogni progresso e chiude la sessione in maniera sicura, riportando l'utente alla schermata di login.

2.7 Use Case 7: Gestione della Persistenza dei Dati

Attore Principale: Sistema (in background)

Descrizione: Il sistema deve salvare e recuperare i dati degli utenti, degli esercizi e dei risultati.

Flusso Principale:

1. Durante ogni operazione (login, registrazione, aggiornamento dei progressi), il FileManager (o Repository) esegue operazioni di lettura/scrittura su file.
2. Il sistema esegue controlli di consistenza e gestisce eventuali errori di I/O.

2.8 Use Case 8: Funzionalità Extra (Opzionali)

Attori Principali: Utente e/o Amministratore

Descrizione: Includere funzionalità aggiuntive per aumentare il valore dell'app.

Esempi di Flusso:

1. **Notifiche:** Il sistema invia notifiche periodiche via email o Telegram riguardo ai progressi o al completamento degli esercizi.
2. **Leaderboards:** Viene visualizzata una classifica con i migliori punteggi, permettendo all'utente di confrontare i risultati con altri.
3. **Modalità Amministratore:** Un utente con privilegi particolari può accedere a un pannello di controllo per aggiungere o modificare esercizi e gestire gli utenti.
4. **Multi-Lingua:** L'utente può selezionare la lingua dell'interfaccia in base alle proprie preferenze.

3 Avvio dell'Applicazione

3.1 Caricamento Iniziale

– L'applicazione si avvia caricando le risorse necessarie e verificando la presenza di eventuali dati utente salvati (file di configurazione, backup, ecc.).

3.2 Schermata Splash (opzionale)

– Visualizzazione di un logo o di un'animazione breve che introduce il progetto.

4 Autenticazione e Registrazione

4.1 Schermata di Login

– **Input:** Username e password.

– **Validazione:** Controllo dei dati inseriti e gestione degli errori (messaggi di errore in caso di credenziali errate).

– **Persistenza:** Lettura dei dati da file (es. in formato JSON o XML) e confronto con quelli inseriti.

4.2 Creazione Nuovo Utente

– **Accesso alla Registrazione:** Pulsante "Crea Nuovo Utente".

– **Form Registrazione:** Inserimento di username, password, nome, cognome e (eventualmente) recapiti per notifiche.

– **Salvataggio:** Scrittura dei dati su file, con controlli di consistenza e gestione degli errori.

4.3 Controlli di Sicurezza

– Validazione dei dati e gestione dei casi di input non valido.

– Possibile crittografia della password prima del salvataggio.

5 Dashboard e Menu Principale

5.1 Visualizzazione della Dashboard

– Griglia degli Esercizi: Icone e mini-descrizioni per ciascuna tipologia di esercizio (es. "Leggi il Codice", "Trova l'Errore", "Cosa Stampa?").

– Barre di Avanzamento: Indicatori grafici che mostrano il progresso e il grado di completamento di ciascun esercizio.

– Accesso a Funzionalità Extra: Link o pulsanti per leaderboard, impostazioni multilingua, notifiche, ecc.

5.2 Navigazione Intuitiva

– Struttura semplice e chiara per passare da un modulo all'altro, con un menu laterale o una barra superiore che consente di accedere rapidamente alle varie sezioni (esercizi, impostazioni, profilo, logout).

6 Selezione e Descrizione dell'Esercizio

6.1 Scelta dell'Esercizio

– L'utente seleziona un esercizio dalla griglia.

6.2 Schermata di Descrizione

– **Informazioni Dettagliate:** Descrizione dell'esercizio, motivazione formativa, regole di esecuzione e criteri per il superamento.

– **Visualizzazione del Livello:** Indicazione del grado attuale (principiante, intermedio, avanzato) e dei requisiti per passare al livello successivo.

– **Opzioni:** Pulsante "Inizia" per avviare l'esercizio o "Annulla" per tornare al menu principale.

7 Esecuzione dell'Esercizio

7.1 Avvio dell'Esercizio

– Caricamento Dinamico: L'applicazione carica dinamicamente il contenuto specifico dell'esercizio scelto.

7.2 Interazione

– Modalità Interattiva: L'utente risponde alle domande/compiti tramite interfaccia grafica (clic, input da tastiera, drag-and-drop, ecc.).

– Feedback Immediato: Possibilità di ricevere feedback su ogni risposta (corretta o errata), con suggerimenti in caso di errori.

7.3 Gestione dei Livelli di Difficoltà

– Progressione: Dopo aver superato un certo numero di esercizi (es. 3/5 risposte corrette consecutivamente), l'app passa automaticamente al livello di difficoltà successivo.

– Interruzione: Il pulsante "Esci dall'esercizio" permette di terminare la sessione in qualsiasi momento, registrando il tentativo come fallito se non completato.

7.4 Implementazioni Aggiuntive

- **Timer:** Un cronometro che misura il tempo impiegato per completare l'esercizio.
- **Backup Progressivo:** Salvataggio periodico dei progressi durante l'esercizio per evitare perdite in caso di chiusura improvvisa.
- **Gestione degli Errori:** Validazione continua degli input, con messaggi di avviso e opzioni di correzione.

8 Risultati e Feedback

8.1 Schermata di Riepilogo

- Risultati dell'Esercizio: Punteggio ottenuto, numero di risposte corrette/errate, livello raggiunto.
- Feedback Dettagliato: Commenti e suggerimenti per migliorare, evidenziando punti di forza e aree di miglioramento.

8.2 Salvataggio dei Risultati

- Persistenza: I dati (nome utente, esercizio, risultato, livello) vengono salvati su file in maniera sicura e consistente.
- Storico dei Risultati: Possibilità di visualizzare uno storico dei tentativi e dei progressi nel tempo.

9 Funzionalità Extra per Ottenere il Voto più Alto

9.1 Notifiche e Comunicazioni

- Invio Automatico: Integrazione con email o Telegram per inviare notifiche con i risultati dell'esercizio.
- Recap Settimanale: Invio di un report settimanale sui progressi e le aree da migliorare.

9.2 Modalità Amministratore

- Gestione Esercizi: Funzionalità che permettono all'amministratore di aggiungere, modificare o eliminare esercizi.
- Gestione Utenti: Possibilità di visualizzare e gestire gli account utente, con statistiche di utilizzo e progressi.

9.3 Leaderboards (Classifiche)

- Ranking: Visualizzazione di una classifica con i punteggi migliori, per stimolare la competizione tra utenti.
- Filtri e Ricerca: Possibilità di filtrare per tipologia di esercizio, livello o periodo.

9.4 Multi-Lingua

– Localizzazione: Implementazione di un sistema per cambiare la lingua dell'interfaccia in base alle preferenze dell'utente.

9.5 Backup e Ripristino Dati

– Automatizzato: Salvataggi periodici e opzioni di ripristino in caso di errori o perdite di dati.

10 Logout e Uscita

10.1 Funzione di Logout

– Salvataggio Finale: Prima del logout, l'applicazione assicura che tutti i progressi e i dati siano salvati correttamente.

– Conferma Uscita: Messaggio di conferma dell'uscita e possibilità di rientrare in una sessione precedente tramite il salvataggio dello stato.

10.2 Chiusura Sicura

– Rilascio Risorse: Pulizia della memoria e chiusura dei file aperti.

– Messaggio di Addio: Schermata finale che ringrazia l'utente e offre eventualmente suggerimenti per ulteriori esercizi o aggiornamenti.

11 Best Practices e Considerazioni Finali

11.1 Documentazione e Commenti

– Il codice deve essere ben documentato con commenti esplicativi in ogni modulo, facilitando la comprensione e la manutenzione futura.

11.2 Testing

– Implementare test unitari e funzionali per ogni componente dell'applicazione, garantendo una copertura completa del codice e una gestione degli errori robusta.

11.3 User Experience (UX)

– Garantire un'interfaccia intuitiva e responsive, con una navigazione fluida e feedback immediato per le azioni dell'utente.

11.4 Consistenza e Robustezza

– Assicurarsi che ogni modulo interagisca in modo consistente con gli altri, con controlli di validità e salvataggio sicuro dei dati.

12 Architettura MVVM

12.1 Model

- **Entità e Classi di Dominio:**
  - **User:** Contiene proprietà come username, password (idealmente crittografata), nome, cognome e lo storico dei progressi (esercizi completati, livelli raggiunti, punteggi).
  - **Exercise (classe astratta):** Definisce i metodi e le proprietà comuni a tutti gli esercizi (titolo, descrizione, livello di difficoltà, regole di esecuzione).
  - **Sottoclassi di Exercise:** ReadCodeExercise, FindErrorExercise, WhatPrintsExercise (o altre tipologie che intendi implementare) con regole specifiche per ciascun tipo.
  - **ExerciseResult:** Registra il risultato di un esercizio, includendo dati quali il punteggio, il tempo impiegato, il numero di risposte corrette/errate e il livello raggiunto.
- **Persistenza e Gestione dei Dati:**
  - **Repository o FileManager:** Una classe che gestisce la lettura e la scrittura dei dati su file (utilizzando JSON, XML, CSV o un formato custom).
  - **Controlli di Consistenza ed Error Handling:** Validazione dei dati in ingresso/uscita e gestione degli errori (es. file non trovato, errori di I/O).

12.2 View

- La View si occupa della presentazione grafica, definendo l'interfaccia utente. Utilizza FXML e JavaFX per creare le scene.
- **Schermate Principali:**
  - **Login/Register:** Layout per l'autenticazione e la creazione di nuovi account.
  - **Dashboard/Griglia degli Esercizi:** Una schermata principale che mostra le icone degli esercizi, barre di avanzamento e le opzioni di navigazione.
  - **Dettaglio Esercizio:** Schermata che mostra la descrizione dell'esercizio, le regole e il livello corrente.
  - **Esecuzione dell'Esercizio:** Layout interattivo in cui l'utente risponde alle domande; possono essere utilizzati componenti come `TextField`, `Button`, `ProgressBar` e timer visivi.
  - **Risultati e Feedback:** Una schermata di riepilogo con il punteggio, feedback e statistiche.
  - **Opzioni Extra:** Se implementi funzionalità opzionali, ad esempio per l'amministratore, per le leaderboard o per la scelta della lingua.
- **Elementi di Interazione:**
  - Componenti UI che rispondono al data binding (label, bottoni, liste, slider, ecc.) e sono progettati per essere semplici, intuitivi e reattivi.

12.3 ViewModel

- Il ViewModel è il mediatore tra Model e View, gestendo il binding e la logica di presentazione.
- **LoginViewModel:**
  - Proprietà: Username, password, messaggi di errore.
  - Comandi/Metodi: Funzioni per effettuare il login e per gestire il recupero o la registrazione dei dati utente (interagendo con il Model).
- **RegisterViewModel:**
  - Proprietà: Dati del nuovo utente (username, password, nome, cognome, ecc.).
  - Comandi: Metodo per validare e salvare i dati di registrazione tramite il FileManager.
- **DashboardViewModel:**
  - Proprietà: Lista degli esercizi (tipicamente un `ObservableList`), proprietà per le barre di avanzamento, dati relativi al profilo utente.
  - Metodi: Funzioni per aggiornare il progresso, per navigare verso il dettaglio di un esercizio e per richiedere i dati aggiornati dal Model.
- **ExerciseViewModel:**
  - Proprietà: Dettagli dell'esercizio (descrizione, regole, livello attuale), stato corrente dell'esercizio.
  - Metodi: Avvio dell'esercizio, gestione del flusso (es. "inizia", "annulla", "esci"), aggiornamento dinamico dello stato in risposta alle risposte dell'utente.
- **ExerciseExecutionViewModel:**
  - Proprietà: Domande correnti, risposte inserite, punteggio in tempo reale, timer (se implementato).
  - Metodi: Valutazione delle risposte, progressione tra i livelli di difficoltà e salvataggio parziale dei progressi.
- **ResultsViewModel:**
  - Proprietà: Dati del risultato (punteggio, numero di tentativi, tempo impiegato).
  - Metodi: Elaborazione dei dati di feedback, possibilità di visualizzare lo storico dei risultati.
- **Comandi e Binding:**
  - Utilizzare le proprietà osservabili (ad es. `ObservableValue`, `SimpleStringProperty`, `SimpleIntegerProperty`, ecc.) per sincronizzare i dati tra il Model e la View in tempo reale.
  - Definire comandi (azioni) eseguibili dalla View (ad es. `loginCommand`, `startExerciseCommand`) che modificano le proprietà esposte nel ViewModel.

13 Integrazione e Best Practices

13.1 Navigazione

- Implementare un sistema di navigazione (ad esempio, un *NavigationManager*) che, coordinato dal ViewModel principale, gestisce il passaggio tra le varie scene in modo fluido e centralizzato.

13.2 Testing

- I ViewModel devono essere testati separatamente dalla UI. Creare test unitari per verificare la logica di binding e i metodi che manipolano i dati.

13.3 Documentazione e Commenti

- Commentare il codice in ogni componente, specificando la funzione di ciascun metodo e il flusso dei dati.

13.4 Gestione degli Errori

- Assicurarsi che sia il Model che il ViewModel abbiano una robusta gestione degli errori, in modo da fornire messaggi utili all'utente e garantire che il sistema sia sempre in uno stato consistente.

13.5 Conclusioni sull'Architettura MVVM

- Utilizzando questo schema MVVM si otterrà una struttura ben separata in cui:
  - Il **Model** gestisce i dati e la logica di business.
  - La **View** è responsabile della presentazione, mantenendo il codice UI separato dalla logica.
  - Il **ViewModel** fa da ponte, gestendo la comunicazione e il binding tra Model e View, semplificando il testing e la manutenzione.