Progetto 2025 - Corso di Programmazione Java Avanzata

# 1. Introduzione

1.1 Scopo del Documento

Il presente documento ha lo scopo di descrivere in dettaglio l’analisi, la progettazione e le scelte tecniche adottate per lo sviluppo dell’applicazione “Progetto 2025”, un software per la visualizzazione temporizzata di documenti e generazione di quiz a risposta multipla, sviluppato in JavaFX.

# 1.2 Descrizione Generale del Sistema

L’applicazione permette all’utente di selezionare un livello di difficoltà e leggere documenti scelti casualmente dal database in base a tale difficoltà per un tempo limitato. Successivamente, l’utente risponde a un quiz generato automaticamente basato sul documento appena letto. Il sistema supporta la gestione utenti, la memorizzazione dei punteggi e dispone di un pannello amministratore per caricare documenti e stopwords.

# Analisi dei Requisiti

## Requisiti Funzionali (RF)

* **RF1: Visualizzazione documenti temporizzata**
* **RF1.1:** L'applicazione deve mostrare uno o più documenti all'utente.
* **RF1.2:** Il tempo di visualizzazione dei documenti deve essere limitato e configurato in base alla difficoltà selezionata.
* **RF1.3:** La difficoltà influisce sulla durata totale di visualizzazione
* **RF1.4:** Al termine del tempo, la visualizzazione termina automaticamente e si passa alla fase successiva (es. quiz).
* **RF2: Generazione domande a risposta multipla**
* **RF2.1:** Il sistema deve generare domande basate sul contenuto dei documenti visualizzati dall’utente e presenti nel database
* **RF2.2:** Le domande devono avere risposte multiple (4 opzioni).
* **RF2.3:** Le risposte devono includere una sola opzione corretta.
* **RF2.4:** Le domande devono variare in base alla difficoltà selezionata (es. domande più complesse con difficoltà maggiore).
* **RF4: Salvataggio punteggi**
* **RF4.1:** Ogni sessione di gioco deve generare un punteggio in base alle risposte corrette e alla difficoltà.
* **RF4.2:** Il punteggio deve essere salvato in un database relazionale.
* **RF4.3:** I punteggi devono essere associati all’utente autenticato.
* **RF5: Pannello amministratore**
* **RF5.1:** Deve essere disponibile un pannello riservato agli amministratori.
* **RF5.2:** L’amministratore può caricare nuovi documenti da inserire nel sistema.
* **RF5.3:** L’amministratore può caricare e gestire liste di stopwords (parole da ignorare nel processo di generazione delle domande).
* **RF5.4:** Il sistema deve garantire la sicurezza e l’accesso limitato a questa funzione solo agli amministratori autorizzati.
* **RF6: Supporto multiutente con tracking punteggi**
* **RF6.1:** Il sistema deve consentire la contemporanea gestione di più utenti.
* **RF6.2:** Ogni utente ha uno storico dei punteggi salvati.
* **RF6.3:** Il sistema deve permettere il recupero e la visualizzazione dello storico punteggi (salvati nel db) per ogni utente.
* **RF7: Visualizzazione leaderboard e statistiche post-gioco (opzionale)**
* **RF7.1:** Il sistema può mostrare una classifica globale basata sui punteggi degli utenti.
* **RF7.2:** Sono disponibili statistiche aggregate (es. media punteggi, punteggi migliori, numero di partite giocate).
* **RF7.3:** L’utente può consultare queste informazioni al termine della sessione di gioco.
* **RF8: Supporto documenti multilingua (opzionale)**
* **RF8.1:** Il sistema deve poter gestire documenti in lingue diverse.
* **RF8.2:** L’utente può selezionare la lingua dei documenti.
* **RF8.3:** La generazione delle domande deve adattarsi alla lingua selezionata.
* **RF9: Ripresa delle sessioni interrotte (opzionale)**
* **RF9.1:** Il sistema deve permettere all’utente di riprendere una sessione di gioco interrotta.
* **RF9.2:** Lo stato della sessione (documenti mostrati, domande già fatte, punteggio corrente) deve essere salvato periodicamente.
* **RF9.3:** L’utente può scegliere di continuare o ricominciare una sessione interrotta.

## Requisiti Non Funzionali (RNF)

**RNF1: Utilizzo di JavaFX**

* **RNF1.1:** L’interfaccia grafica deve essere sviluppata utilizzando il framework JavaFX.
* **RNF1.2:** Devono essere sfruttate le caratteristiche di JavaFX per la gestione delle scene degli eventi.

**RNF2: Utilizzo librerie standard JDK 8 o superiore**

* **RNF2.1:** Non devono essere utilizzate librerie esterne o di terze parti non incluse nel JDK standard.
* **RNF2.2:** Tutte le funzionalità devono essere implementate con le API offerte dal JDK (esclusi tool esterni).

**RNF3: Uso di Stream API e lambda expression**

* **RNF3.1:** Il codice deve fare ampio uso di Stream API per operazioni su collezioni (filtri, mappe, riduzioni).
* **RNF3.2:** Devono essere usate lambda expression per rendere il codice conciso e leggibile.
* **RNF3.3:** Queste caratteristiche devono essere presenti in almeno il 70% dei moduli di business logic.

**RNF4: Styling con CSS**

* **RNF4.1:** L’interfaccia deve essere stilizzata tramite file CSS esterni.
* **RNF4.2:** I temi grafici devono essere configurabili senza modificare il codice Java.
* **RNF4.3:** Lo stile deve essere coerente e supportare diverse risoluzioni e dimensioni di finestra.

**RNF5: Interfaccia user-friendly e reattiva**

* **RNF5.1:** L’interfaccia deve essere intuitiva e semplice da usare anche per utenti non esperti.
* **RNF5.2:** Le azioni dell’utente devono ricevere un feedback immediato (es. messaggi, animazioni).
* **RNF5.3:** L’applicazione deve mantenere tempi di risposta rapidi (es. caricamento schermate entro 1 secondo).
* **RNF5.4:** L’app deve essere utilizzabile in finestre ridimensionabili senza perdita di funzionalità o leggibilità.

# Matrice di Tracciabilità Requisiti-Casi d'uso

| Requisito | Descrizione | Stato | Caso d'Uso Associato | Note |
| --- | --- | --- | --- | --- |
| RF1 | Visualizzazione documenti temporizzata | ✅ Accettato | CU1 | Implementato con timer e scelta difficoltà |
| RF2 | Generazione domande a risposta multipla | ✅ Accettato | CU2 | Domande auto-generate dal testo |
| RF3 | Registrazione e autenticazione utente | ✅ Accettato | CU3 | Con validazione e hash password |
| RF4 | Salvataggio punteggi | ✅ Accettato | CU4 | DB con punteggio associato alla sessione |
| RF5 | Caricamento documenti e stopwords (Admin) | ✅ Accettato | CU5 | Pannello amministratore disponibile |
| RF6 | Tracking punteggi multiutente | ✅ Accettato | CU4 | Salvataggio e recupero per utente |
| RF7 | Statistiche e leaderboard post-gioco *(opz.)* | ✅ Accettato | CU6 | Funzionalità base realizzata |
| RF8 | Supporto a più lingue *(opz.)* | ❌ Non Accettato | CU7 | Non implementato nel prototipo attuale |
| RF9 | Ripresa sessioni interrotte *(opz.)* | ❌ Non Accettato | CU8 | Non implementato, da valutare per versioni future |

# Casi d'uso principali

**CU1 – Visualizzazione documenti in base alla difficoltà selezionata**

* **Attori:** Utente autenticato
* **Descrizione:**  
  L’utente sceglie un livello di difficoltà, dopodiché l’app mostra uno o più documenti per un tempo limitato, variabile in base al livello.
* **Precondizioni:**  
  L’utente è autenticato. Sono presenti documenti nel sistema.
* **Flusso principale:**
  1. L’utente seleziona la difficoltà.
  2. Il sistema carica i documenti relativi.
  3. Viene avviato un timer (es. 30 sec per difficoltà "facile").
  4. Alla scadenza del tempo, i documenti non sono più visibili.
* **Estensioni:**
  1. Se il documento è mancante → messaggio d’errore e ritorno alla selezione.
* **Postcondizioni:**  
  L’utente ha letto i documenti e può rispondere alle domande.
* **Requisiti collegati:** RF1, RNF5

----------------------------------------------------------------------------------------------------------------------

**CU2 – Generazione automatica delle domande a risposta multipla**

* **Attori:** Sistema, Utente autenticato
* **Descrizione:**  
  Il sistema genera automaticamente delle domande sulla base dei documenti letti.
* **Precondizioni:**  
  L’utente ha completato la fase di lettura dei documenti.
* **Flusso principale:**
  1. Il sistema analizza il contenuto del documento.
  2. Rimuove le stopwords.
  3. Estrae concetti chiave e genera domande con risposte multiple.
  4. Le domande vengono presentate all’utente.
  5. L’utente risponde.
* **Postcondizioni:**  
  Le risposte vengono valutate per il punteggio.
* **Requisiti collegati:** RF2, RF5, RNF3

----------------------------------------------------------------------------------------------------------------------

**CU3 – Registrazione e login utente**

* **Attori:** Utente
* **Descrizione:**  
  L’utente può creare un account o accedere con credenziali esistenti.
* **Precondizioni:**  
  Nessuna (per la registrazione) / account esistente (per il login).
* **Flusso principale:**
  + Registrazione:
    1. L’utente inserisce username, email e password.
    2. Il sistema verifica validità e unicità.
    3. Crea l’account.
  + Login:
    1. L’utente inserisce credenziali.
    2. Il sistema verifica ed esegue l’accesso.
* **Estensioni:**
  + Dati errati → messaggio d’errore.
  + Utente già registrato → suggerimento login.
* **Postcondizioni:**  
  Utente autenticato e pronto a giocare.
* **Requisiti collegati:** RF3, RNF5

----------------------------------------------------------------------------------------------------------------------

**CU4 – Salvataggio e visualizzazione punteggi**

* **Attori:** Sistema, Utente
* **Descrizione:**  
  Al termine del quiz, il punteggio viene calcolato e salvato nel database.
* **Precondizioni:**  
  L’utente ha completato una sessione.
* **Flusso principale:**
  1. Il sistema calcola il punteggio in base a risposte corrette, tempo, difficoltà.
  2. Salva il punteggio nel DB associato all’utente.
  3. Mostra punteggio e storico all’utente.
* **Postcondizioni:**  
  I dati della sessione sono salvati e accessibili.
* **Requisiti collegati:** RF4, RF6, RNF2

----------------------------------------------------------------------------------------------------------------------

**CU5 – Caricamento documenti e stopwords da parte dell’amministratore**

* **Attori:** Amministratore
* **Descrizione:**  
  Un pannello consente all’amministratore di caricare nuovi documenti e liste di stopwords.
* **Precondizioni:**  
  L’utente ha privilegi di amministrazione.
* **Flusso principale:**
  1. Admin accede al pannello.
  2. Seleziona un file (documento o stopword list).
  3. Il sistema carica e valida il file.
  4. I dati vengono memorizzati nel database.
* **Estensioni:**
  1. File non valido → messaggio d’errore.
* **Postcondizioni:**  
  I documenti sono disponibili per l’uso.
* **Requisiti collegati:** RF5

----------------------------------------------------------------------------------------------------------------------

**CU6 – Visualizzazione leaderboard e statistiche (opzionale)**

* **Attori:** Utente autenticato
* **Descrizione:**  
  Al termine di una sessione, l’utente può visualizzare statistiche e classifiche.
* **Precondizioni:**  
  L’utente ha completato almeno una sessione.
* **Flusso principale:**
  1. L’utente accede alla sezione leaderboard.
  2. Il sistema mostra i punteggi migliori e medie.
  3. Mostra anche i dati dello storico personale.
* **Requisiti collegati:** RF7

----------------------------------------------------------------------------------------------------------------------

**CU7 – Selezione e caricamento documenti in lingua specifica (opzionale)**

* **Attori:** Utente, Sistema
* **Descrizione:**  
  L’utente seleziona una lingua e il sistema carica documenti corrispondenti.
* **Precondizioni:**  
  Documenti in lingua disponibili.
* **Flusso principale:**
  1. L’utente seleziona la lingua.
  2. Il sistema filtra i documenti in base alla lingua.
  3. Carica i documenti nella lingua selezionata.
* **Requisiti collegati:** RF8

----------------------------------------------------------------------------------------------------------------------

**CU8 – Ripresa delle sessioni interrotte (opzionale)**

* **Attori:** Utente autenticato
* **Descrizione:**  
  Se una sessione si interrompe, l’utente può riprenderla da dove era rimasto.
* **Precondizioni:**  
  Sessione interrotta esistente salvata.
* **Flusso principale:**
  1. Il sistema rileva sessioni interrotte.
  2. Chiede all’utente se desidera riprendere.
  3. Ricarica stato precedente (documenti, tempo, domande risposte).
* **Postcondizioni:**  
  L’utente può completare la sessione.
* **Requisiti collegati:** RF9

# Scelte Progettuali e Architetturali

**1. Architettura Generale: MVC (Model-View-Controller)**

L'applicazione sarà strutturata seguendo il pattern **MVC**, utile per separare:

* **Model:** Logica applicativa e accesso ai dati (utenti, documenti, sessioni, punteggi).
* **View:** Interfaccia grafica realizzata in **JavaFX + CSS**.
* **Controller:** Gestione delle interazioni tra utente e logica, uso intensivo di **lambda expressions** e **JavaFX events**.

**2. Suddivisione in Packages**

| **Modulo** | **Contenuto** | **Dipendenze** |
| --- | --- | --- |
| alert | Messaggi di errore e avvertimento nel programma | Nessuna |
| controller | Logica di controllo per ogni schermata o processo e accesso al db | alert, view, users |
| view | File FXML e CSS, classi JavaFX legate all’interfaccia | controller |
| service | Funzionalità del programma | alert, users |
| users | Suddivisione dei ruoli degli utenti | Nessuna |

**3. Pattern Software Utilizzati**

| **Pattern** | **Applicazione** | **Motivazione** |
| --- | --- | --- |
| MVC | Architettura generale | Separazione di responsabilità |
| DAO (Data Access Object) | Accesso al DB per utenti, documenti, punteggi | Astrazione e riusabilità |
| Factory | Creazione dinamica di domande in base al tipo | Flessibilità e estendibilità |
| Observer/Listener | Notifica eventi UI (es. Timer countdown) | Gestione eventi asincroni |
| Singleton | Gestione sessione utente autenticato | Condivisione stato globale controllato |

**4. Persistenza dei dati**

**Tecnologie**: SQLite

**Accesso**: DAO pattern

**Tabelle principali:**

* Documents (id, text, title, difficulty)
* Sessions (username, score, timestamp, id, difficulty, id\_document)
* Users (username, password, role)
* Words (id\_document, word, frequency)

**5. Interfaccia Utente (UI)**

**Tecnologia**: JavaFX

**Struttura**: FXML per la struttura (lo styling CSS è all’interno dei file .fxml)

**Comportamento dinamico**: Binding, animazioni , reattività tramite eventi

**Schermate**:

* Login e Registrazione
* Menù di gioco
* Lettura documento con countdown
* Quiz a risposta multipla
* Statistiche e leaderboard
* Profilo dell’utente

**6. Sicurezza**

**Sessione utente:** Singleton nella classe GameSessions

**Controlli UI:** Visibilità di alcune schermate e accesso a servizi differenti vincolati al ruolo utente (utente/admin)

**Coesione ed Accoppiamento delle Classi**

**(controller)**

**AdminController – Coesione Funzionale**

* DocumentsManagement (service): Accoppiamento per controllo
* Alert (javafx): Accoppiamento per controllo
* LoadDocumentView.fxml (view): Accoppiamento aree comuni

**DocumentReadController – Coesione Sequenziale**

* DatabaseManagement (service): Accoppiamento per dati

**LevelsController – Coesione Funzionale**

* DocumentReadView.fxml, MainMenu.fxml(view): Accoppiamento aree comuni

**LoadDocumentDialogController – Coesione Funzionale**

* DocumentsManagement, Levels (service): Accoppiamento per controllo
* AdminController: Accoppiamento per contenuti

**LoginController – Coesione Funzionale**

* Login (service): Accoppiamento per controllo
* Player (users): Accoppiamento per dati
* Role (users): Accoppiamento per timbro
* GameSessionManagement (service): Accoppiamento per controllo
* MainMenu.fxml/AdminView.fxml: Accoppiamento per aree comuni

**MainApp**

* GameSessionManagement (service): Accoppiamento per controllo

**MainMenuController – Coesione Funzionale**

* GameSessionManagement (service): Accoppiamento per controllo
* GameView.fxml: Accoppiamento per aree comuni
* Level.fxml : Accoppiamento per aree comuni
* UserManagementView.fxml : Accoppiamento per aree comuni
* PersonalScoresView.fxml: Accoppiamento per aree comuni

**PersonalScoresViewController – Coesione Comunicazionale**

Opera sugli stessi dati delle sessioni ma con operazioni diverse in quanto è predisposta per visualizzare i dati

* GameSessionManagement, GameSession(service): Accoppiamento di controllo
* MainMenu.fxml, PersonalScoreView.fxml (view): Accoppiamento aree comuni

**QuestionsController – Coesione Funzionale**

* GameSessionManagement (service): Accoppiamento per controllo
* DocumentReadView.fxml, ResultsView.fxml (view); Accoppiamento aree comuni

**ResultsController – Coesione Funzionale**

* MainMenu.fxml, PersonalScoresView.fxml, DocumentReadView.fxml (Aree Comuni)
* GameSessionManagement (service): Accoppiamento per dati

**UserManagementViewController – Coesione Funzionale**

* Player (users): Accoppiamento per dati
* GameSessionManagement (service): Accoppiamento per controllo
* MainMenu.fxml (view): Accoppiamento per aree comuni

**(users)**

**User – Coesione Funzionale**

**Admin – Coesione Comunicazionale** perchè sfrutta gli stessi dati dello User (ereditati) più il campo “specializzante”

**Player – Coesione Logica** poichè sfrutta due campi, Role e PersonalScore (che potrebbe avere una classe a parte)

**Role – Coesione Funzionale** (anche se non sono presenti metodi poichè è un tipo enum)

(service)

DataBaseManagement – Coesione Funzionale

* Alert : accoppiamento per controllo

DocumentsManagement – Coesione Logica perchè combina diverse responsabilità (db, documenti e filtri per I testi, che potrebbe essere suddiviso in un modulo a parte “ProcessoreTesti” e “DocumentoDBManager)

* DataBaseManagement, Levels : Accoppiamento per controllo

GameSession – Coesione Funzionale

* Nessun accoppiamento

GameSessionManagement – Coesione Comunicazionale perchè I metodi currentPlayer, score e timestamp lavorano sugli stessi dati ma in maniera differente)

* DataBaseManagement, Levels: Accoppiamento per controllo
* Player (users) : Accoppiamento per controllo

Levels – Coesione Funzionale

* Nessun accoppiamento

Login – Coesione Funzionale

* DataBaseManagement: Accoppiamento per dati
* Player, Role (users) : Accoppiamento per dati (poichè crea degli oggetti)

Question – Coesione Funzionale

* Nessun accoppiamento

QuestionGenerator – Coesione Funzionale

* DataBaseManagement, Word, Question: Accoppiamento per dati

UsersManagement – Coesione comunicazionale poichè sfrutta direttamente users

* DataBaseManagement: Accoppiamento per dati

Word – Coesione Funzionale

* Nessun accoppiamento