WORDAGEDDON: SFIDA LA TUA MEMORIA!

Obiettivi e Tecnologie Chiave

Obiettivi Principali

- Sviluppare un sistema di gaming educativo per la memoria verbale.
- Implementare algoritmi di analisi testuale efficienti con Java Stream API.
- Creare un'interfaccia utente intuitiva e responsiva con JavaFX.
- Gestire un sistema multiutente con persistenza dati su database relazionale.
- Fornire funzionalità amministrative per la gestione dei contenuti.

Java 8

Ampio utilizzo di Stream API e programmazione funzionale.

JavaFX

Framework GUI per un'interfaccia utente moderna.

JDBC

Connessione a sistemi di gestione database relazionali.

Pattern MVC

Architettura Model-View-Controller per modularità.

Requisiti Funzionali e Non Funzionali

Requisiti Funzionali

- **Gestione Sessioni di Gioco:** Configurazione con parametri variabili, visualizzazione documenti a tempo, generazione domande multiple.
- **Tipologie di Domande:** Frequenza assoluta, associazione documento-specifica, esclusione etc...
- **Gestione Utenti:** Registrazione, autenticazione, associazione punteggi, persistenza dati.
- **Funzionalità Amministrative:** Caricamento documenti, gestione stopwords.
- **Funzionalità Avanzate:** Classifiche globali/personali, statistiche post-gioco, salvataggio sessioni.

Requisiti Non Funzionali

Performance

Analisi documenti in tempo reale.

Usabilità

Interfaccia intuitiva e responsive.

Scalabilità

Supporto per documenti di grandi dimensioni.

Manutenibilità

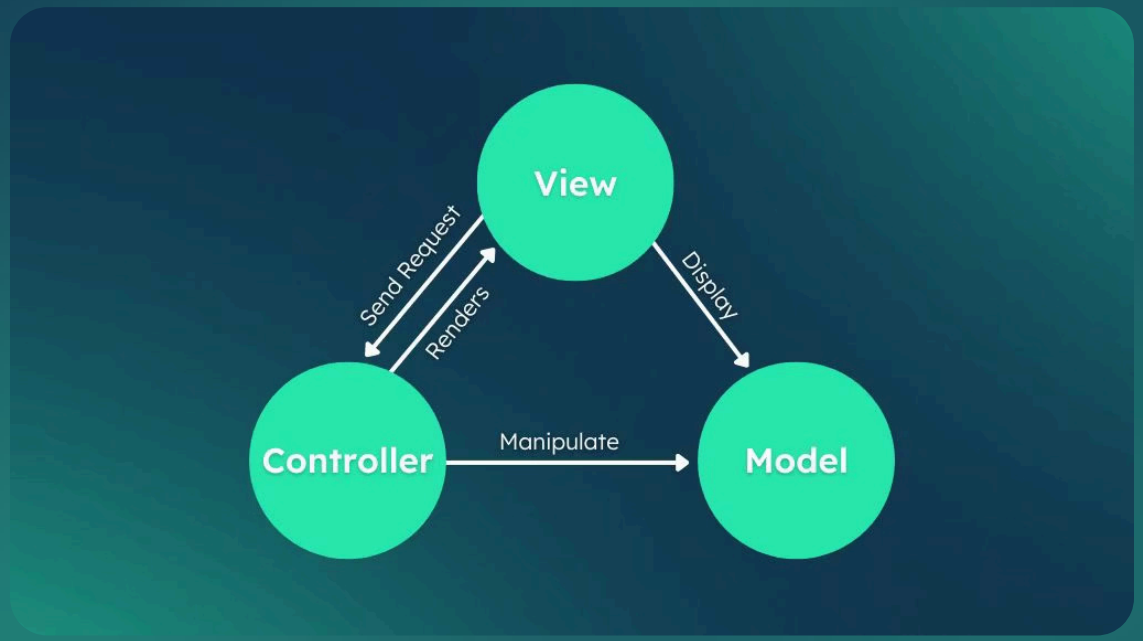
Codice ben strutturato e documentato.

Architettura del Sistema: Model-View-Controller

Il sistema adotta un'architettura a tre livelli per un codice ordinato e modulare: Presentation Layer (JavaFX), Business Logic Layer (gestione gioco, analisi testi) e Data Access Layer (database/file di salvataggio sessione).

L'applicazione segue il pattern Model-View-Controller (MVC), che facilita manutenzione ed espansione. Il Model contiene i dati fondamentali (Utente, Sessione, Documento, Domanda). La View è costituita dal file FXML per l'interfaccia utente. Il Controller gestisce l'interazione tra vista e modello, coordinando i cambiamenti dell'interfaccia.

Questa organizzazione a pacchetti (model, view, controller) rende chiara la responsabilità di ogni componente, migliorando modularità, testabilità e riutilizzo del codice.



Implementazione: Analisi Testuale e Generazione Domande

Analisi Testuale

Preprocessing

Rimozione punteggiatura e normalizzazione case.

Tokenizzazione

Suddivisione in token tramite regex.

Filtering

Rimozione stopwords (lato query SQL).

Frequency Analysis

Calcolo frequenze.

Generazione Domande

Il sistema seleziona casualmente una delle quattro tipologie di domande disponibili nel database. Per ogni domanda, viene scelto un documento di riferimento e una parola chiave. Le risposte sono generate casualmente, coerentemente con la logica della tipologia specifica.

1. **Frequenza Massima Assoluta:** Calcolo della parola più frequente in tutti i documenti.
2. **Frequenza relativa:** Calcolo delle occorrenze di una parola in un documento specifico.
3. **Esclusione:** Identificazione parole assenti, cioè non presenti in nessun documento
4. **Frequenza Massima Relativa:** Calcolo della parola più frequente in un documento specifico

Schema del database

Il sistema utilizza un database PostgreSQL per persistenza.

Lo schema del database è progettato per efficienza. Supporta la gestione delle partite e l'analisi testuale.

Users	Memorizza credenziali dei giocatori.
Sessione	Registra i dettagli di ogni partita giocata.
Documento	Contiene i testi usati per generare domande.
Domanda	Definisce i template delle domande
Mappa	Mantiene l'analisi del contenuto di ogni documento
StopWords	Mantiene tutte le stopwords caricate dall'utente
Vocabolario	Mantiene parole casuali , utili per poter mantenere la risposta corretta alla domanda : "Qual è la parola che non si presenta in nessun documento"

Scelte cruciali: Logiche di Gioco e Salvataggio

Generazione Dinamica Domande

La generazione domande è una scelta chiave. Si basa su quattro template con selezione casuale di documento e parola chiave. Offre una sfida sempre nuova.

Salvataggio e Ripristino Sessione

La sessione viene salvata automaticamente in un file `.dat`. Il salvataggio avviene solo se la partita è incompleta. Ciò permette di riprendere il gioco in seguito.

Risultati e Conclusioni

Funzionalità Implementate

- Sistema di gioco multilivello completamente funzionale.
- Quattro tipologie di domande con generazione automatica.
- Pannello amministrativo per gestione contenuti.
- Funzionalità avanzate: classifiche, statistiche.

Qualità del Codice

- Ampio utilizzo di Stream API e programmazione funzionale.
- Architettura modulare con basso accoppiamento.
- Documentazione Javadoc completa.
- Gestione eccezioni robusta e logging strutturato.