Gruppo 10
Gregorio Barberio, Francesco Peluso, Davide Quaranta, Ciro Ronca

Introduzione

Wordageddon è un'applicazione JavaFX concepita come un **gioco educativo** volto a stimolare la memoria a breve termine e le capacità di analisi testuale degli utenti.

A partire da uno o più documenti visualizzati per un tempo limitato, il giocatore è chiamato a rispondere a una serie di domande a risposta multipla basate su **statistiche testuali**, come frequenze di parole, confronti e assenze nei documenti.

Fase di progettazione

Requisiti e vincoli di progetto

Requisiti funzionali

ID	Requisito	Descrizione
RF1	Visualizzazione documenti	L'utente visualizza uno o più documenti testuali per un tempo limitato.
RF2	Selezione difficoltà	I parametri di gioco variano in base alla difficoltà scelta (facile, medio, difficile).
RF3	Generazione domande	Al termine della lettura vengono generate domande a risposta multipla basate sui documenti.
RF4	Tipologie di domande	Le domande includono frequenze, confronti, esclusioni e riferimenti a documenti specifici.

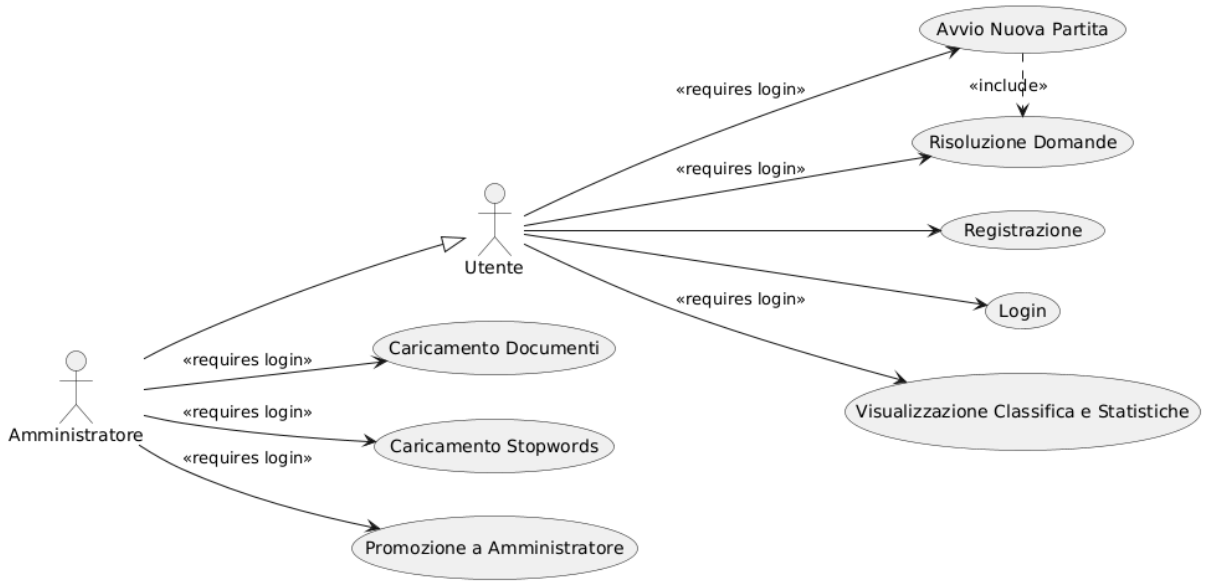
RF5	Registrazione/Login	Il sistema permette la creazione di account e autenticazione utenti.
RF6	Salvataggio punteggio	Ogni sessione produce un punteggio salvato e associato all'utente nel database.
RF7	Pannello amministratore	L'admin può caricare nuovi documenti e liste di stopwords.
RF8	Supporto multilingua (opzionale, non implementata ancora)	È possibile caricare documenti in più lingue e selezionare quella desiderata.
RF9	Leaderboard	Classifica globale o personale dei punteggi ottenuti dagli utenti.
RF10	Statistiche post-gioco	Dopo la partita si mostrano errori, parole frequenti e confronto risposte.
RF11	Ripresa sessioni (opzionale, non implementata ancora)	Le sessioni possono essere salvate e riprese in un secondo momento.

Requisiti non funzionali

ID	Requisito	Descrizione
RNF1	Tecnologie consentite	Solo uso di JDK 8+ senza librerie esterne (tranne JDBC).
RNF2	Interfaccia grafica	Da implementare con framework JavaFX (incluso in JDK8 o caricato esternamente per JDK 9+) con (facoltativamente) personalizzazione con stylesheet CSS.
RNF3	Persistenza dati	Viene richiesto di implementare la persistenza dei dati per quanto riguardano le principali entità di interesse nel gioco.
RNF4	Usabilità	Il sistema deve essere facilmente utilizzabile anche da utenti non esperti.
RNF5	Prestazioni	L'analisi dei documenti deve essere efficiente anche su testi lunghi, e per le operazioni critiche (es. fase di gameplay) i tempi di caricamento devono essere minimi.

Gruppo 10
Gregorio Barberio, Francesco Peluso, Davide Quaranta, Ciro Ronca

Diagramma dei casi d'uso



Nota: *si sono omessi alcuni dettagli nello schema al fine di rendere questo più leggibile. Si fa presente che per la generazione dei diagrammi è stato utilizzato il tool* ***PlantUML****, e che i file in alta qualità dei diagrammi sono disponibili nella cartella della consegna del progetto.*

Descrizione dei casi d'uso

UC1 – Registrazione Utente

- **Attore primario:** Utente
- **Precondizioni:**
  - L'utente non è ancora registrato nel sistema.
  - Ha accesso alla piattaforma.
- **Postcondizioni:**
  - L'account viene creato e registrato nel database.
  - L'utente può accedere alle funzionalità del gioco.
- **Flusso principale:**
  - L'utente accede alla schermata di registrazione.
  - Inserisce nome, cognome, email e password.
  - Il sistema verifica la validità dei dati.
  - Il sistema registra il nuovo account.
- **Flussi alternativi:**
  - Dati incompleti o non validi: il sistema mostra un messaggio di errore.
  - Email già registrata: viene richiesto di inserirne un'altra.

UC2 – Login Utente

- **Attore primario:** Utente
- **Precondizioni:**
  - L'utente è registrato.
  - È in possesso delle credenziali.
- **Postcondizioni:**
  - Accesso alla dashboard personale.
- **Flusso principale:**
  - L'utente inserisce email e password.
  - Il sistema verifica le credenziali.
  - Se corrette, avviene l'accesso.
- **Flusso alternativo:**
  - Credenziali errate: il sistema mostra un messaggio di errore e consente di riprovare.

UC3 – Avvio Nuova Partita

- **Attore primario:** Utente
- **Precondizioni:**
  - L'utente è autenticato.
  - Il sistema è pronto per una nuova sessione.
- **Postcondizioni:**
  - Avvio di una nuova sessione di gioco.
  - Il punteggio viene inizializzato.
- **Flusso principale:**
  - L'utente seleziona il livello di difficoltà.
  - Il sistema seleziona i documenti in base al livello.
  - I documenti vengono mostrati e parte il timer di lettura.
  - L'utente legge i testi entro il tempo limite.
- **Flusso alternativo:**
  - Se il tempo scade, il sistema passa automaticamente alla fase del quiz.

UC4 – Risoluzione Domande

- **Attore primario:** Utente
- **Precondizioni:**
  - La lettura dei documenti è completata.
  - Le domande sono pronte.
- **Postcondizioni:**
  - Le risposte sono salvate per il calcolo del punteggio.
  - Viene mostrato un resoconto con i risultati.
- **Flusso principale:**
  - Il sistema presenta una domanda con quattro opzioni.
  - L'utente seleziona una risposta.
  - Si procede con la domanda successiva fino al termine.
  - Il sistema mostra il riepilogo con indicazione delle risposte corrette ed errate.
- **Flusso alternativo:**
  - L'utente interrompe la partita e torna al menu.

UC5 – Caricamento Documenti (amministrazione)

- **Attore primario:** Amministratore
- **Precondizioni:**
  - L'amministratore è autenticato.
  - I documenti sono pronti per il caricamento.
- **Postcondizioni:**
  - I documenti vengono salvati nel sistema e saranno disponibili per il gioco.
- **Flusso principale:**
  - L'amministratore accede al pannello di gestione.
  - Seleziona la funzione "Carica Documento".
  - Sceglie un file da caricare.
  - Il sistema salva il file nel database.
  - Il sistema mostra una conferma.
- **Flusso alternativo:**
  - Il file selezionato è vuoto o non valido: viene mostrato un messaggio di errore.

UC6 – Caricamento Stopwords (amministrazione)

- **Attore primario:** Amministratore
- **Precondizioni:**
  - L'amministratore è autenticato.
- **Postcondizioni:**
  - La lista di stopwords viene aggiornata nel sistema.
- **Flusso principale:**
  - Accesso alla sezione gestione stopwords.
  - Caricamento del file contenente la lista.
  - Il sistema verifica il formato del file.
  - Il file viene caricato nel database.
  - Il sistema conferma l'avvenuto caricamento.
- **Flusso alternativo:**
  - Il file è malformato o contiene errori: viene richiesto di riprovare.

UC7 – Visualizzazione Classifica e Statistiche

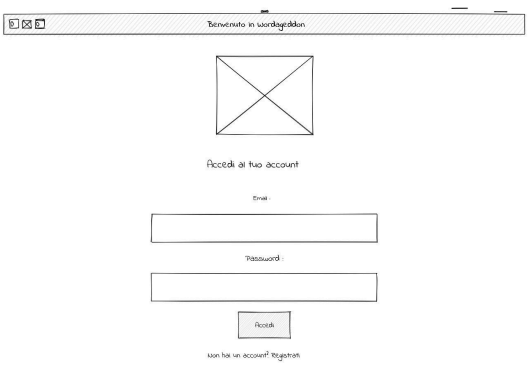
- **Attore primario:** Utente
- **Precondizioni:**
  - L'utente ha già partecipato ad almeno una sessione di gioco.
- **Postcondizioni:**
  - Vengono visualizzate la classifica e le statistiche personali.
- **Flusso principale:**
  - L'utente accede alla sezione classifica.
  - Il sistema recupera i dati dal database.
  - Viene mostrata la classifica e, se disponibili, le statistiche individuali.
- **Flusso alternativo:**
  - Nessun punteggio registrato: viene mostrato un messaggio informativo.

UC8 – Promozione a Amministratore

- **Attore primario:** Amministratore
- **Precondizioni:**
  - L'utente da promuovere è registrato nel sistema.
  - L'operazione è eseguita da un amministratore autenticato.
- **Postcondizioni:**
  - L'utente acquisisce i privilegi amministrativi.
- **Flusso principale:**
  - L'amministratore accede al pannello utenti.
  - Seleziona l'utente da promuovere.
  - Convalida la promozione.
  - Il sistema aggiorna il ruolo nel database.
  - Il sistema notifica il buon esito dell'operazione.
- **Flusso alternativo:**
  - L'utente è già amministratore: il sistema avvisa che l'operazione non è necessaria.
  - L'utente che tenta la promozione non ha i privilegi necessari: viene mostrato un messaggio di errore.

Mockup tramite wireframes

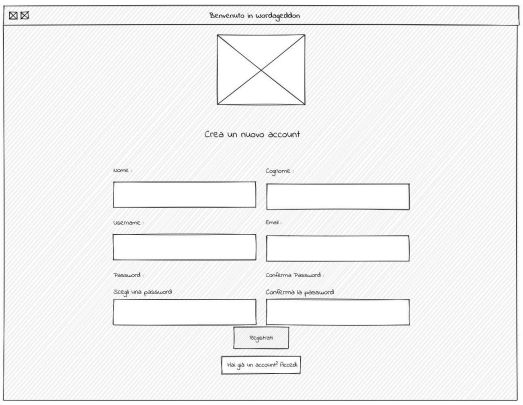
Di seguito sono rappresentati i wireframe delle varie schermate che sono state poi implementate in FXML tramite SceneBuilder, e che sono state inoltre personalizzate con dei link a degli stylesheet CSS.



Schermata login

Schermata iniziale dell'applicazione. L'utente viene invitato ad effettuare il login tramite la propria email e password, o, se non si ha un account, ad effettuare la registrazione.

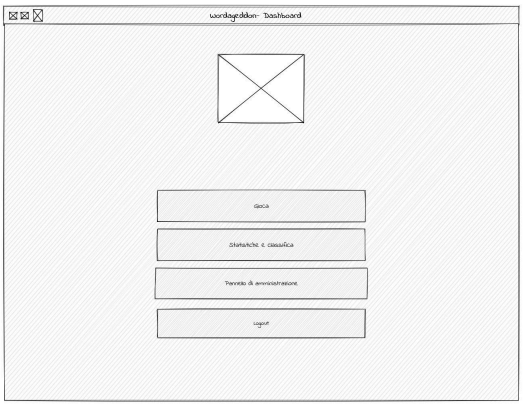
Gruppo 10
Gregorio Barberio, Francesco Peluso, Davide Quaranta, Ciro Ronca



Schermata registrazione

All'utente viene chiesto di inserire il suo nome e cognome, e di scegliere uno username, inserire il proprio indirizzo email, e di inserire la password (con conferma).

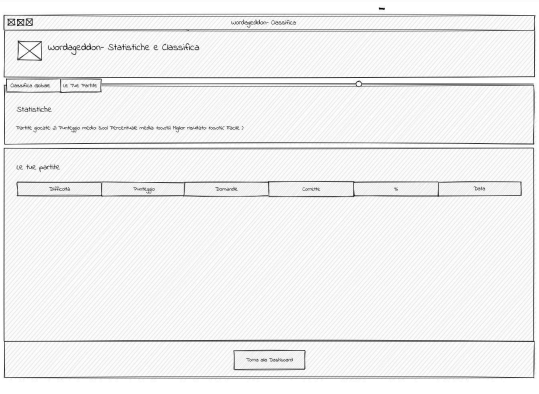
Effettuata la registrazione, l'utente sarà reindirizzato al login.

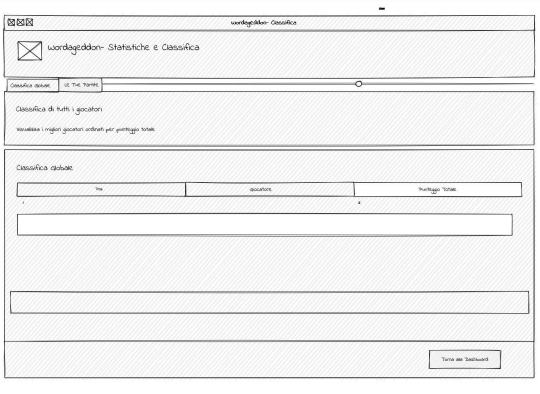


Schermata registrazione

Menù principale del gioco. Sono disponibili quattro scelte:

- effettuare una partita
- visualizzare statistiche partite e classifica
- (se admin) accedere alle impostazioni
- effettuare il logout (reindirizza a login)



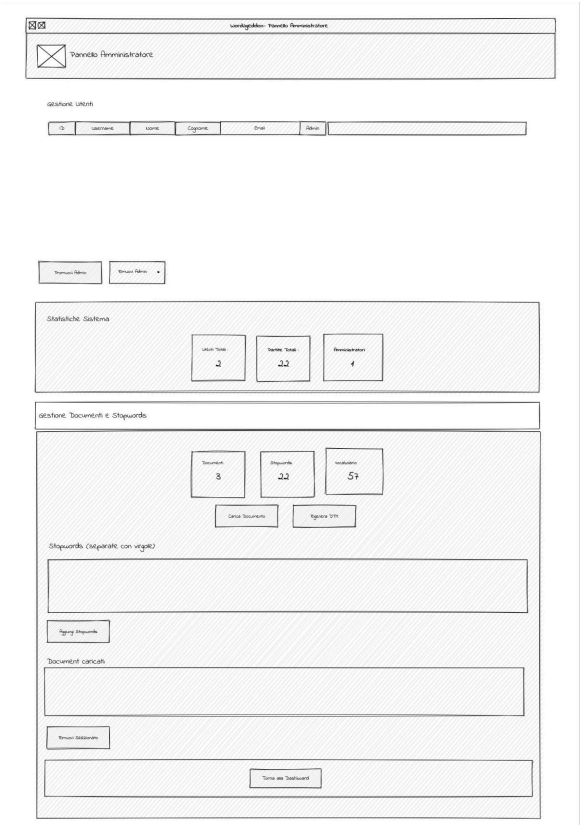


Schermata statistiche e classifica

Schermata divisa in due tab: *classifica globale con tutti i giocatori* e *elenco partite effettuate*.

L'utente ha visione dei punteggi degli altri giocatori, e dei punti che ha totalizzato (assieme ad alcune statistiche) dalle partite effettuate.

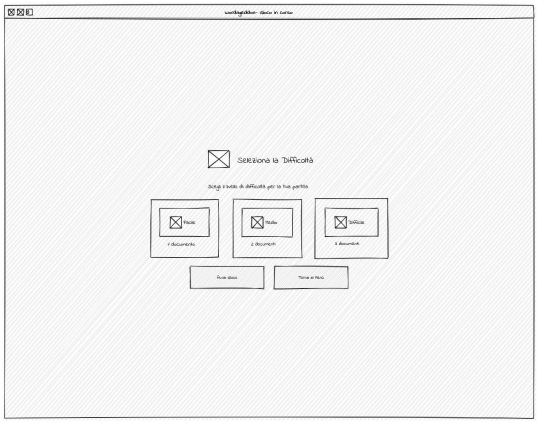
Gruppo 10
Gregorio Barberio, Francesco Peluso, Davide Quaranta, Ciro Ronca



Schermata impostazioni

Questa schermata è accessibile solo dagli utenti con privilegi di amministratore.

Da qui è possibile visualizzare la lista degli utenti registrati con la possibilità di garantire o revocare i permessi di amministrazione; vedere alcune statistiche di sistema riguardanti i giocatori e le partite; visualizzare statistiche e gestire i documenti caricati assieme alle stopwords, ovvero i termini di cui non si vuole avere traccia della document term matrix.

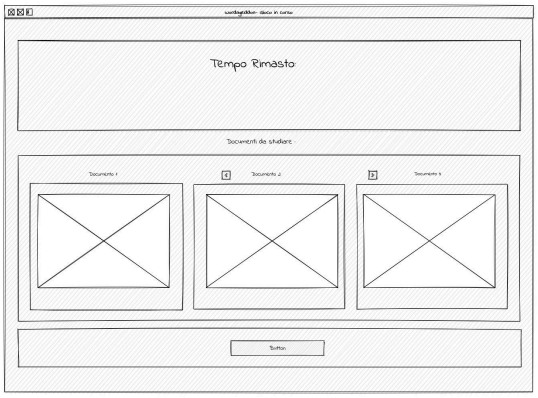


Schermata inizio partita

All'utente che fa richiesta di iniziare una nuova partita, viene chiesto il grado di difficoltà che si vuole affrontare per questa sessione di gioco.

Documentazione Javadocs: https://francescopeluso.github.io/AdvancedJava-Project

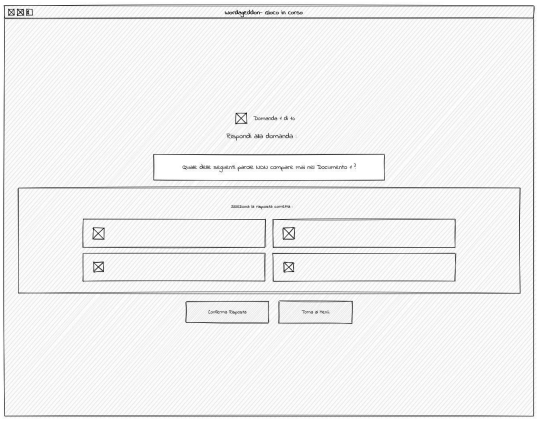
Gruppo 10
Gregorio Barberio, Francesco Peluso, Davide Quaranta, Ciro Ronca



Schermata lettura documenti

Iniziata la partita, l'utente in 30 secondi dovrà leggere da 1 a 3 documenti (in base alla difficoltà scelta).

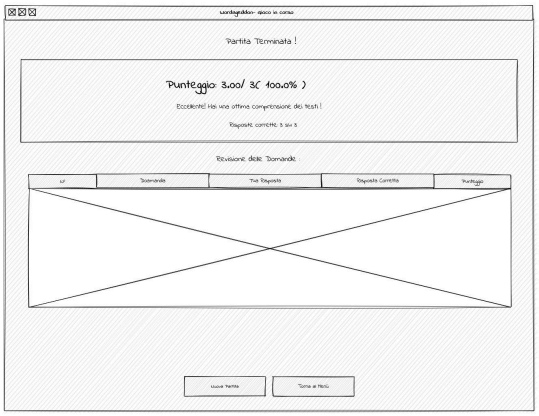
Può scegliere di annullare tutto e tornare al menù, oppure aspettare che cominci la fase di risposta alle domande (dopo 10s potrà direttamente avviare la seconda fase).



Schermata di risposta ai quiz

Dopo la fase di lettura, l'utente dovrà rispondere a 3, 5 o 10 domande generate casualmente dal sistema selezionando una delle quattro opzioni che saranno disponibili a schermo.

Anche qui può scegliere di proseguire o di annullare la sessione di gioco.



Schermata riepilogo partita

Terminato il quiz, all'utente verrà mostrato il punteggio totalizzato, e potrà fare una revisione delle risposte date alle domande che gli sono state poste sui documenti.

Qui può scegliere di tornare al menù, oppure di avviare una nuova partita (verrà reindirizzato alla scelta della difficoltà).

Gruppo 10
Gregorio Barberio, Francesco Peluso, Davide Quaranta, Ciro Ronca

Descrizione del gameplay

Date le caratteristiche che il gioco deve rispettare, abbiamo deciso di strutturare il gioco in questa maniera:

- ogni sessione di gioco è caratterizzato da un livello di difficoltà:
  - facile - 1 documento con 3 domande,
  - medio - 2 documenti con 5 domande,
  - difficile - 3 documenti con 10 domande;

- la fase di lettura dei documenti dura 30 secondi; i primi 10 secondi non possono essere saltati (si potrà tornare solo al menù principale) - superati i primi 10 secondi, non appena si sente pronto l'utente può decidere di passare alla fase successiva del gameplay (almeno che non scada il tempo, dove in quel caso il passaggio alla fase successiva verrà forzato);

- le domande poste all'utente sono generate sulla base dei documenti visualizzati, e sono delle seguenti tipologie (hardcoded nel codice, non è possibile aggiungerne altre se non si modifica il codice sorgente del progetto):
  - domande di frequenza assoluta,
  - domande di confronto,
  - domande per documento specifico,
  - domande di esclusione;

- in base alla risposta:
  - se corretta, allora l'utente guadagna 1 punto,
  - se sbagliata, allora l'utente perde -0.33 punti,
  - il totale potrà dunque essere negativo, il che influirà sulla sua posizione in classifica globale;

- in base a tutte le sessioni di gioco affrontate dagli utenti, grazie all'implementazione della persistenza dei dati su database è possibile visualizzare una classifica globale in cui ognuno potrà vedere i punti totalizzati dagli altri giocatori.

Gruppo 10
Gregorio Barberio, Francesco Peluso, Davide Quaranta, Ciro Ronca

Stack tecnologico

Il progetto è stato scritto in **Java**, utilizzando la **JDK versione 8**.

L'utilizzo di questa versione della JDK ci permette di sfruttare alcuni meccanismi ed API viste durante il corso - tra queste, nel progetto spiccano l'utilizzo delle *lambda expressions*, *method references*, *Reflection API* (usato principalmente in funzioni che interagiscono col framework grafico).

Purtroppo, poiché non disponibili in questa versione, non è stato fatto uso di ulteriori caratteristiche presenti nelle più moderne versioni di Java (es. JDK 17) come le *sealed classes* e i *record type*.

Per la realizzazione dell'interfaccia grafica è stato usato il framework **JavaFX**, le cui librerie, per quanto riguarda il nostro caso, sono già incluse all'interno del development kit.

Come richiesto dal requisito non funzionale **RFN1**, non abbiamo fatto utilizzo di ulteriori librerie di terze parti, tranne per quanto riguarda il **Java Database Connector SQLite (JDBC)** per permettere alla nostra applicazione di implementare la persistenza dei dati su un database relazionale **SQLite**.

Gruppo 10
Gregorio Barberio, Francesco Peluso, Davide Quaranta, Ciro Ronca

Dettagli implementativi

Tutto il funzionamento del programma e del gameplay gira tutto intorno ad una struttura nota come **Document Term Matrix**, che *è una matrice matematica che descrive la frequenza dei termini che si verificano in ogni documento di una collezione.*

Questa matrice è stata implementata tramite due **Hash Maps** annidate, in modo da garantire una complessità computazionale in lettura/scrittura pari, nel caso medio, a O(1), dunque tempi di accesso e scrittura estremamente rapidi.

La prima (esterna) ha set di keys i nomi dei documenti caricati dall'utente, mentre la seconda (interna, dunque value di ogni value associata ad ogni key della map esterna) ha come set di chiavi i vari token rilevati nei documenti (*esclusi quelli nel set di stopwords*), a quali sono associati come valori le relative frequenze nel documento.

Ovviamente abbiamo deciso di rendere la matrice persistente, ipotizzando che con il passare del tempo un utente avrebbe caricato un numero crescente di documenti con cui interagire; calcolare la matrice ogni volta sarebbe diventato progressivamente più dispendioso.

Inoltre, l'analisi dei documenti di testo viene effettuata per mezzo delle Stream API, che ci forniscono un metodo rapido ed efficace per processare grandi flussi di testo in modo funzionale e scalabile, applicando filtri, trasformazioni e aggregazioni in tempo reale senza dover caricare l'intero documento in memoria

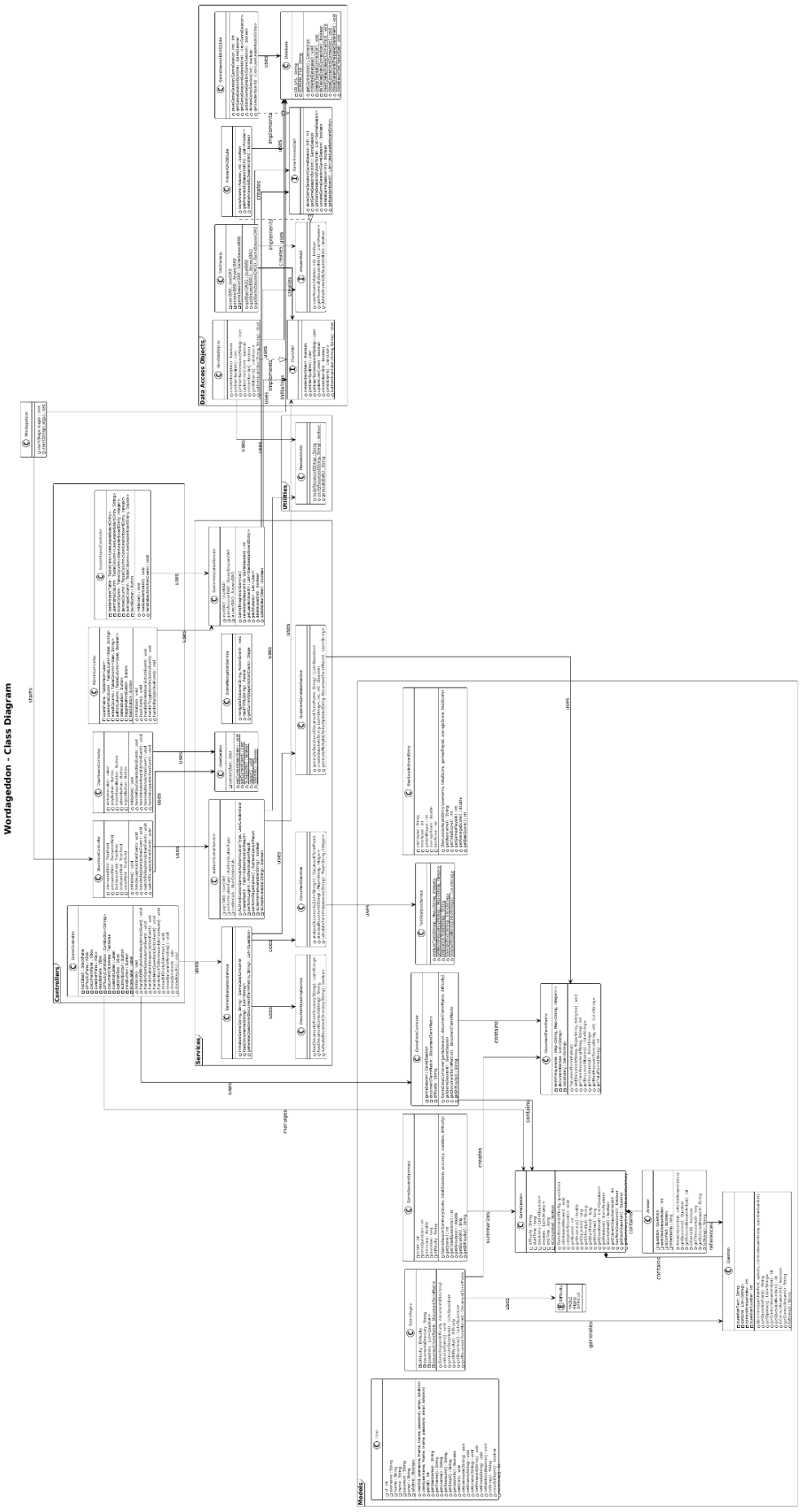
Classi utilizzate

Nella prossima pagina è riportato il **diagramma delle classi**, dove queste ultime sono raggruppate per **package**, posizionata di traverso per migliorare la leggibilità di questa.

Si fa presente che per la generazione dei diagrammi è stato utilizzato il tool **PlantUML**, e che i file in alta qualità dei diagrammi sono disponibili nella cartella della consegna del progetto.

La descrizione delle classi è stata omessa da questo documento, poiché sono disponibili **online** le **JavaDocs complete** del progetto (*link a piè di pagina*).

Gruppo 10
Gregorio Barberio, Francesco Peluso, Davide Quaranta, Ciro Ronca

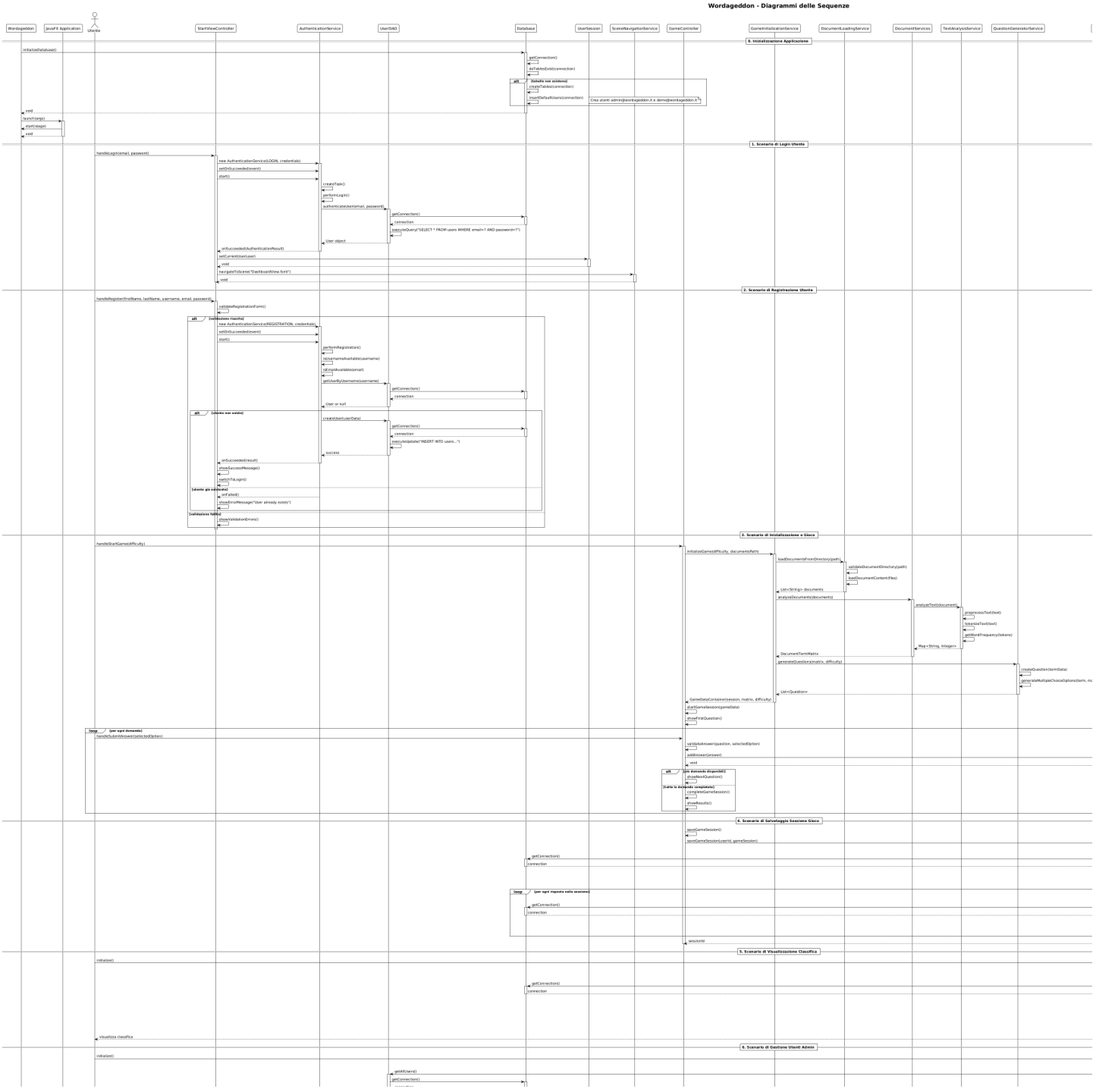


Documentazione Javadocs: https://francescopeluso.github.io/AdvancedJava-Project

Gruppo 10
Gregorio Barberio, Francesco Peluso, Davide Quaranta, Ciro Ronca

Diagramma delle sequenze

Qui viene riportato il diagramma delle sequenze. Si fa presente che per la generazione dei diagrammi è stato utilizzato il tool **PlantUML**, e che i file in alta qualità dei diagrammi sono disponibili nella cartella della consegna del progetto. (**l'immagine sotto è tagliata**)



Documentazione Javadocs: https://francescopeluso.github.io/AdvancedJava-Project

Gruppo 10
Gregorio Barberio, Francesco Peluso, Davide Quaranta, Ciro Ronca

Persistenza dei dati

Database

Il progetto Wordageddon implementa un sistema di persistenza dei dati utilizzando il pattern **Data Access Object (DAO)** combinato con l'utilizzo di **SQLite3** come database relazionale.

L'architettura adottata garantisce una chiara separazione tra la logica di business e l'accesso ai dati, facilitando la manutenibilità e l'estensibilità del sistema per possibili ampliamenti delle funzionalità che si vorranno aggiungere.

Come si può vedere anche dalla struttura dei package e delle classi, siamo andati a creare interfaccia ed implementazione per ogni entità di cui ci interessava tenere traccia sul database.

I pattern progettuali seguiti per l'implementazione della persistenza su database sono:

1. **Data Access Object (DAO)**
   - ➢ incapsula l'accesso ai dati fornendo un'interfaccia uniforme
2. **Factory Pattern**
   - ➢ gestisce la creazione e l'accesso ai DAO
3. **Singleton Pattern**
   - ➢ garantisce una sola istanza per ogni DAO
4. **Connection Pool**
   - ➢ gestione ottimizzata delle connessioni al database

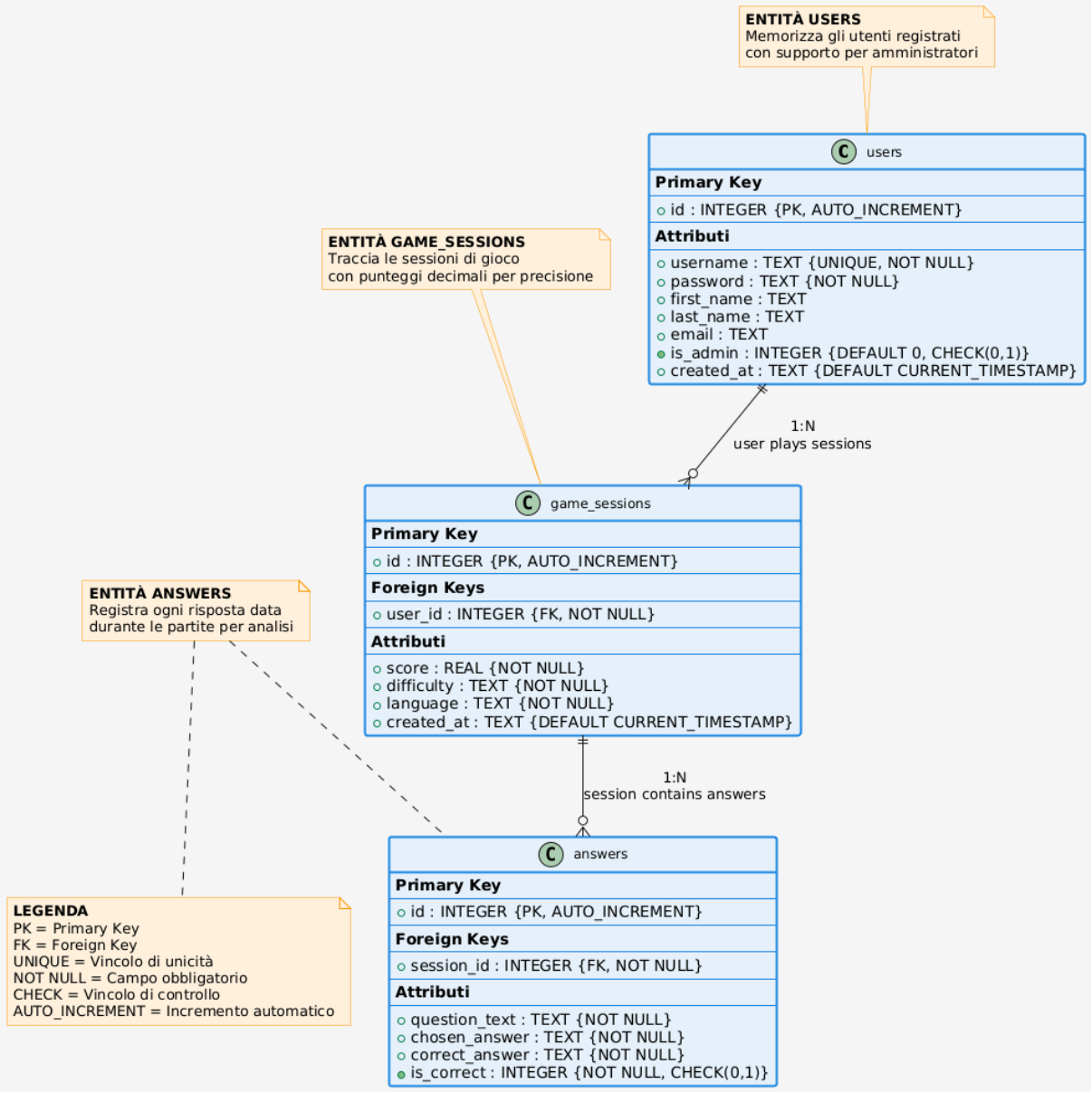
Le tre entità di cui abbiamo rilevato avere interesse nel tenere traccia sul database sono:

- **Utenti**
- **Sessioni di gioco effettuate**
- **Risposte alle domande di una sessione**

Nota: *per motivi di sicurezza, è stata implementata una classe* ***PasswordUtils*** *che permette l'hashing e la verifica delle password al login tramite algoritmo SHA-256 con salting.*

Nella pagina seguente viene mostrato lo schema logico-concettuale di come è stato poi realizzato il database schema. Il tutto risulta essere normalizzato, e soprattutto sono stati creati vincoli di integrità inter-referenziale, in modo da garantire coerenza tra i dati ed evitare la generazione di tuple spurie ed informazioni inesatte.

Gruppo 10
Gregorio Barberio, Francesco Peluso, Davide Quaranta, Ciro Ronca



Files

Oltre alla persistenza relazionale tramite database SQLite, il progetto implementa un sistema di persistenza su file per gestire dati complessi e garantire prestazioni ottimali.

Questo sistema utilizza principalmente:

- serializzazione Java per oggetti complessi,
- l'I/O su file di testo per documenti.

La principale entità di cui si voleva avere la persistenza su file, in modo da garantire durante l'utilizzo dell'applicazione le migliori performance possibili, sono la **Document Term Matrix**, l'**insieme di stopwords** inserite dall'utente nel pannello amministrazione, e la **lista di documenti** che sono stati caricati.

Per semplicità, queste tre entità sono state incapsulate sotto un'unica classe chiamata **GameDataContainer**, che implementa l'interfaccia *Serializable* e che semplicemente implementa setter e getter per i tre oggetti incapsulati.

Gruppo 10
Gregorio Barberio, Francesco Peluso, Davide Quaranta, Ciro Ronca

JavaFX tasks and services

Il progetto implementa un'architettura “orientata ai servizi” che separa la logica di business dall'interfaccia utente, utilizzando il pattern **Service Layer** per gestire operazioni asincrone e lunghe elaborazioni.

Tutti i servizi estendono la classe **javafx.concurrent.Service** o **javafx.concurrent.Task** per garantire operazioni non bloccanti sul thread principale della UI di JavaFX.

Tra questi abbiamo:

- **AuthenticationService**
  - Scopo: Gestisce login e registrazione utenti
  - Funzionalità: Autenticazione con hash password, validazione credenziali
  - Risultato: *AuthenticationResult* con stato successo/errore

- **DocumentLoadingService**
  - Scopo: Carica e processa documenti di testo in background
  - Funzionalità: Selezione casuale documenti, creazione DTM
  - Risultato: *DocumentLoadingResult* con DTM e contenuti

- **DocumentServices**
  - Scopo: Gestione documenti e persistenza dati su file
  - Funzionalità: Serializzazione game_data.ser, gestione directory data/documents/
  - Caratteristiche: “Caching dei dati”, integrazione *TextAnalysisService*

- **GameInitializationService**
  - Scopo: Inizializza componenti di gioco
  - Funzionalità: Caricamento stopwords, creazione TextAnalysisService
  - Risultato: *GameInitializationResult* con servizio pronto

- **GameIntegrationService**
  - Scopo: Integrazione tra logica di gioco e persistenza database
  - Funzionalità: Salvataggio sessioni, recupero statistiche, leaderboard globali

- **QuestionGeneratorService**
  - Scopo: Generazione domande per il gioco
  - Tipi Domande:
    - Frequenza assoluta: "Quante volte appare X?"
    - Confronto frequenze: "Quale parola appare di più?"
    - Associazione documento: "In quale documento appare X?"
    - Esclusione: "Quale parola NON appare?"
    - Caratteristiche: Evita di proporre la stessa coppia di domanda e set di opzioni di risposta

- **SceneNavigationService**
  - Scopo: Navigazione tra scene FXML
  - Funzionalità: Caricamento asincrono FXML, gestione controller
  - Risultato: NavigationResult con scene e controller

- **UserSession (Singleton)**
  - Scopo: Gestione sessione utente globale
  - Funzionalità: setCurrentUser(), getCurrentUser(), isLoggedIn(), logout()
  - Caratteristiche: Thread-safe, stato applicazione

Durante la progettazione e la scrittura delle classi, sono seguiti i principi fondamentali del design di un'applicazione, come *Separation of Concerns*, *DRY Principle*, *Single Responsibility*, ecc…

Tutto ciò, ovviamente, è per rendere il sistema più manutenibile e scalabile, non solo dagli sviluppatori originali ma anche da altri che potrebbero subentrare, grazie anche alla ricca documentazione fornita tramite i JavaDocs e ai commenti nei source files.

Per provare l'applicazione

Quando il database viene inizializzato per la prima volta dal programma, vengono fornite le due utenze di default:

- admin@wordageddon.it (pw: admin123)
- demo@wordageddon.it (pw: demo123)

Assicurarsi di eseguire il programma con la JDK 8.