Piattaforma Interattiva per l'Apprendimento della Programmazione Java in Architettura MVVM con JavaFX

Matteo Nanni Christian Toffalori

17 maggio 2025

1 Istruzioni per avviare il progetto JavaFX con Maven

1. **Apri il progetto**
   - Avvia IntelliJ IDEA.
   - Vai su `File > Open` e seleziona la cartella del progetto (`Programmazione-main`).
2. **Verifica che il progetto sia Maven**
   - Controlla la presenza di `pom.xml` nella root.
   - Se non riconosciuto, fai clic destro su `pom.xml` e seleziona `Add as Maven project`.
3. **Configura SDK e JavaFX**
   - Vai su `File > Project Structure > Project`:
     - Imposta `Project SDK` su Java 17 o superiore.
     - Imposta `Project language level` su almeno 17.
   - Vai su `File > Project Structure > Modules > Dependencies` e verifica l'importazione delle dipendenze Maven.
4. **Crea una configurazione di Run**
   - In alto a destra, clicca su `Add Configuration...`.
   - Premi `+` e scegli `Maven`.
   - Imposta:
     - **Name:** Run JavaFX
     - **Working directory:** radice del progetto
     - **Command line:** `javafx:run`
   - Salva (`Apply > OK`).
5. **Avvia il progetto**
   - Seleziona la configurazione `Run JavaFX` e clicca su `Run`.

2 Introduzione

L'obiettivo di questo progetto è lo sviluppo di un'applicazione desktop per l'apprendimento della programmazione Java, realizzata in JavaFX e progettata per essere estensibile, interattiva e modulare. L'applicazione guida l'utente attraverso esercizi suddivisi per tematiche e livello di difficoltà, adottando l'architettura MVVM.

3 Obiettivi del Progetto

- Facilitare l'apprendimento pratico della programmazione Java.
- Fornire feedback immediato sugli esercizi.
- Memorizzare i progressi localmente.
- Presentare un'interfaccia utente chiara.
- Garantire la scalabilità del progetto nel tempo.

4 Analisi dei Requisiti

4.1 Requisiti Funzionali

- Autenticazione e registrazione utenti.
- Dashboard con 10 macro tipologie di esercizi.
- Ogni tipologia: 3 esercizi facili, 3 intermedi, 3 difficili.
- Modalità a risposta multipla.
- Feedback immediato e registrazione del progresso.

4.2 Requisiti Non Funzionali

- Dati salvati in file locali (`users.json`, `exercises.json`).
- Progetto gestito con Maven (`pom.xml`).

5 Architettura del Sistema

L'applicazione segue il pattern *Model-View-ViewModel* (MVVM):

- **Model:** entità di dominio (`User`, `Exercise`, `Question`, `Progress`, ecc.).
- **ViewModel:** logica di interazione (`LoginViewModel`, `ExercisesViewModel`, `HomeViewModel`, ecc.).
- **View:** definita in `.fxml` e controllata da controller dedicati (`LoginController`, `PathController`, `TopicHomeController`, ecc.).

6 Struttura del Codice e Modularità

Il codice è organizzato in pacchetti Java:

- `model/`: rappresentazioni dei dati.
- `repository/`: gestione persistenza su file.
- `view/`: controller della GUI.
- `viewmodel/`: logica tra vista e modello.
- `util/`: configurazioni globali (`Config.java`).

Questa struttura facilita l'estensibilità: aggiungere nuove tipologie di esercizi richiede modifiche minime.

7 Gestione dei Dati

I dati vengono salvati in JSON nella cartella `src/data/`:

- `users.json`: dati anagrafici, credenziali, progresso utenti.
- `exercises.json`: esercizi per categoria e difficoltà.

I repository leggono/scrivono tramite serializzazione con wrapper (`UsersWrapper`, `ExercisesWrapper`).

8 Build Automation e Gestione Dipendenze

Il progetto utilizza Apache Maven per:

- Gestione automatica delle dipendenze.
- Ciclo di build standard (compile, test, package).
- Uniformità su diverse macchine.

File chiave: `pom.xml`.

9 Interazione con l'Utente

- **Login/Registrazione:** creazione account e accesso ai progressi.
- **Navigazione:** scelta tra le 10 tipologie di esercizi tramite `PathController` e `TopicHomeController`.
- **Esercizi:** presentati in GUI interattiva (`ExercisesController`) con risposte multiple e caselle di input.

10 Diagramma UML

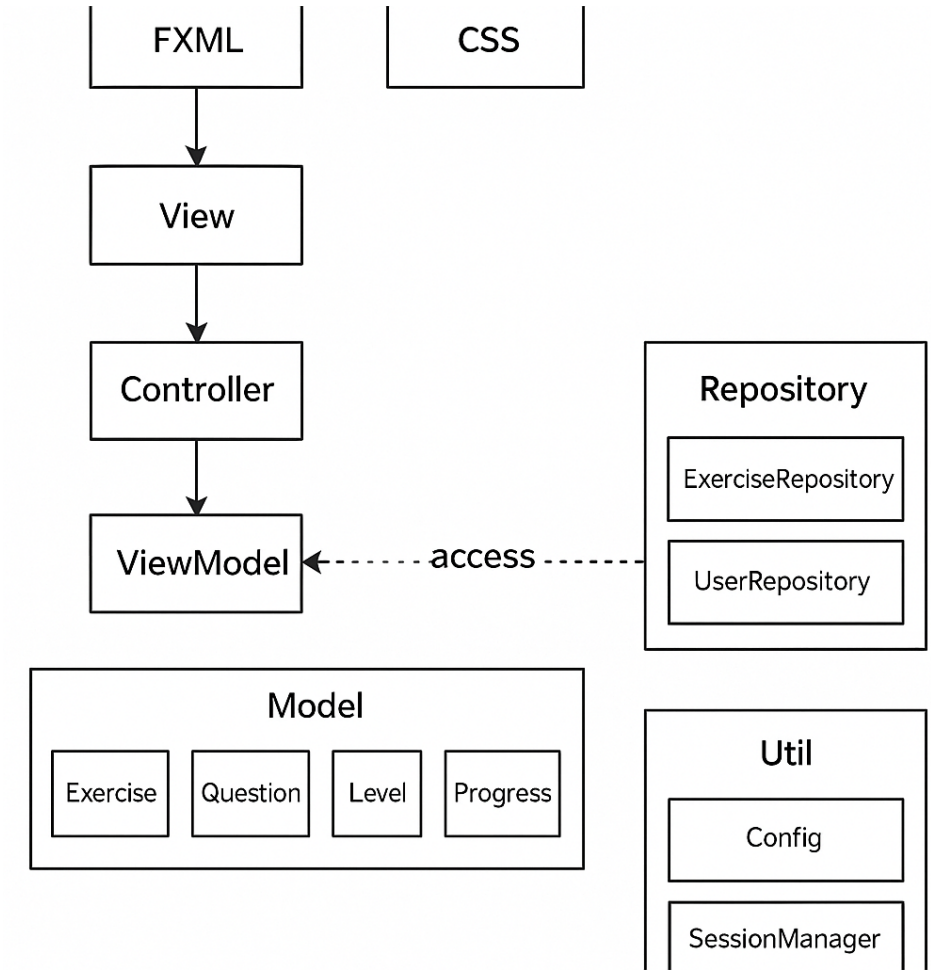

Figura 1: Struttura UML dei pacchetti e delle classi principali.