Specifica dei Requisiti: Sala Giochi Virtuale

Progetto: Sala Giochi Virtuale
Data: 10 Gennaio 2026
Team di Progetto: Team UNI
Corso: Ingegneria del Software – Università degli Studi di Bergamo

Contenuto:

- Requirements Engineering
- Software Quality

1. Introduzione

Il prodotto è un'applicazione desktop *standalone* sviluppata in Java che simula un ambiente di sala giochi. L'obiettivo è fornire una piattaforma di intrattenimento che permetta la gestione utente (registrazione/login), la fruizione di quattro videogiochi classici (Tris, BattleShip, Roulette, DiceGuesser) e la persistenza dei punteggi tramite database locale.

2. Elicitazione dei Requisiti

Il processo di ingegneria dei requisiti è stato condotto dal team seguendo un approccio iterativo tramite un *team working* organizzato, basato sull'analisi del dominio e sui vincoli accademici.

2.1 Metodologia di Elicitazione

I requisiti sono stati raccolti e raffinati attraverso le seguenti tecniche:

1. **Analisi dei Vincoli Accademici (Constraint Analysis):**
    - La fonte primaria dei requisiti è stata la specifica del progetto d'esame. Sono stati identificati i vincoli tecnologici mandatori (uso di *JavaFX*, *Database H2*, uso dei Design Pattern) che hanno costituito la base dei *Requisiti Non Funzionali*.

2. **Brainstorming e Workshop Interni:**
    - Il team ha svolto sessioni di brainstorming (durante la fase di *Analisi e Avvio*) per definire il perimetro funzionale. In particolare, è stata discussa la scelta dei giochi da implementare, valutando il rapporto tra complessità algoritmica, tempo a disposizione e fattibilità.
3. **Analisi degli Scenari (Use Case Scenarios):**
    - Sono stati delineati i flussi di interazione utente-sistema al fine di derivare i requisiti funzionali di dettaglio, la gestione delle eccezioni e dei comportamenti ad alto livello dell'applicazione.

3. Specifica dei Requisiti

3.1 Requisiti Funzionali

Di seguito sono elencate le funzionalità che il sistema deve obbligatoriamente fornire.

- ➢ **RF01 - Registrazione:** Il sistema deve permettere a un nuovo utente di registrarsi inserendo nickname, nome e password.
- ➢ **RF02 - Autenticazione:** Il sistema deve permettere il login tramite nickname e password.
- ➢ **RF03 - Navigazione:** L'utente autenticato deve poter navigare tra la Home, il Tabellone e la selezione dei giochi.
- ➢ **RF04 - Gameplay:** Il sistema deve permettere l'esecuzione di 4 giochi (`Tris`, `BattleShip`, `Roulette`, `DiceGuesser`).
- ➢ **RF05 - Persistenza Punteggi:** Al termine di ogni partita, il sistema deve aggiornare il punteggio dell'utente nel database.
- ➢ **RF06 - Classifica:** Il sistema deve mostrare una classifica globale ordinata per punteggio decrescente.
- ➢ **RF07 - Modalità Anonima:** Il sistema deve permettere l'accesso ai giochi anche senza registrazione, in modalità anonima.

3.2 Requisiti Non Funzionali (Vincoli):

- ➢ **RNF01 - Portabilità:** Il software deve essere esportabile ed eseguibile su qualsiasi macchina con Eclipse avente una JDK aggiornata almeno alla versione 21 senza strumenti esterni all'IDE.
- ➢ **RNF02 - Persistenza Dati:** Il database deve essere di tipo Embedded (H2).
- ➢ **RNF03 - Interfaccia Grafica:** L'interfaccia deve essere realizzata con librerie JavaFX.
- ➢ **RNF04**: L'applicazione deve seguire la traduzione dei diagrammi UML.

3.3 Requisiti Out of Scope

Per chiarezza, si elencano le funzionalità esplicitamente escluse dalla *release* attuale.

1. **Architettura Distribuita**: Il sistema è progettato come applicazione *standalone* a singola istanza. Non sono supportate funzionalità di rete, sincronizzazione remota o interazione tra istanze diverse su macchine separate.
2. **Web Deployment**: L'applicazione è nativa Desktop. Non è prevista un'architettura *Web-Based* né la pubblicazione su *application server* accessibili via internet.
3. **Multimedia e Adattabilità UI**: Il focus del progetto è sulla logica funzionale; pertanto, non sono implementati componenti audio (effetti sonori/musica) e l'interfaccia grafica è a dimensione fissa (non responsive), non adattabile dinamicamente al ridimensionamento della finestra.
4. **Target Dispositivi Mobili**: Il software è ottimizzato esclusivamente per ambienti Desktop (PC/Laptop). Non è previsto il supporto, né l'integrazione, con dispositivi mobili (Smartphone/Tablet) o ambienti Android/iOS; non viene pertanto fornito un eseguibile pre-compilato dell'applicazione.

4. Software Quality (Attributi di Qualità)

Il codice è stato sottoposto a revisione automatizzata tramite **SonarQube, PMD e stan4J**. Questo processo ha permesso di individuare *code smells*, violazioni degli standard Java e potenziali vulnerabilità prima della fase di test dinamico.

Il progetto ha privilegiato inoltre dei seguenti attributi di qualità, ritenuti critici per il contesto accademico e la natura del software:

4.1 Installabilità e Portabilità (Installability)

- **Contesto:** Il software deve essere valutato su macchine diverse da quelle di sviluppo.
- **Implementazione:** L'uso di un Database Embedded (H2) e la gestione delle dipendenze tramite Maven garantiscono che il progetto sia *Self-Contained.* Clonando la repository ed eseguendo la build, il software è immediatamente operativo senza configurazioni di sistema complesse.

4.2 Manutenibilità (Maintainability)

Il software potrebbe richiedere estensioni future (es. nuovi giochi); a tal proposito il software possiede determinate qualità che semplificano l'aggiunta delle suddette estensioni:

- **Modularità:** L'architettura separa nettamente la logica (package `.videogame`), i dati (package `.db`) e la grafica (package `.gui`).
- **Riusabilità:** La classe astratta `VideoGames` permette di aggiungere un nuovo gioco scrivendo solo la logica specifica, ereditando automaticamente metodi comuni come la gestione dei punteggi tra utente e computer.
- **Leggibilità:** Il codice rispetta le Java Coding Conventions standard.

4.3 Usabilità (Usability)

- **Implementazione:**
    - **Learnability:** L'interfaccia è *user-friendly* e utilizza metafore note (es. icone per i giochi, frecce per la navigazione).
    - **Feedback:** Il sistema fornisce feedback immediati (tramite `Alert` e `pop-up`) in caso di errori (es. login fallito) o successo (es. vittoria gioco).

4.4 Affidabilità (Reliability)

Una qualità del software è quella di garantire il mantenimento persistente dei dati tra una sessione e l'altra.

- **Implementazione:**
    - **Fault Tolerance (Parziale):** La classe `DataBaseInitializer` previene il crash dell'applicazione nel caso in cui il file del database venga accidentalmente cancellato o corrotto, rigenerando la struttura necessaria al funzionamento base.

5. Gestione delle Priorità (Metodo MoSCoW)

La tecnica *MoSCoW* è stata essenziale per rispettare le scadenze accademiche senza compromettere la stabilità del sistema, definendo un perimetro chiaro del progetto rispetto ai requisiti concordati.

- **Must-Have:** Rappresentano il *Minimum Viable Product* (MVP). Senza la persistenza dei dati e la logica di gioco, il progetto non soddisferebbe i requisiti minimi.
- **Should-Have:** Funzionalità che aumentano significativamente il valore del software (come la classifica), ma non pregiudicano il funzionamento dei giochi.
- **Could-Have:** Migliorie dell'esperienza utente (modalità anonima), incluse come migliorie aggiuntive ai requisiti base dell’applicazione.
- **Won't-Have:** Funzionalità escluse per limitare l'ambito (scope) e garantire la consegna, come il multiplayer online che richiederebbe una gestione complessa della rete e della concorrenza.

Change Management

Ogni modifica ai requisiti è stata discussa collegialmente durante le sessioni di PP. L'adozione di un'**architettura a tre livelli** (Presentation, Logic, Data Access) ha minimizzato l'impatto dei cambiamenti.