

Project Plan: Sala Giochi Virtuale

Data: 17 novembre 2025

Versione: 2.2 - Correzione riguardo ai requisiti

Team di Progetto: Team UNI

1. Introduzione

Il presente progetto nasce come prova integrativa per il corso di Ingegneria del Software dell'Anno Accademico 2025/2026 presso l'Università degli Studi di Bergamo.

L'obiettivo formativo primario era applicare i principi di progettazione e sviluppo software alla creazione di un'applicazione desktop tramite un team working organizzato, prendendo spunto da strutture apprese durante il corso (integrazione del capo programmatore con il *Pair Programming*).

Il progetto ha messo il team nella condizione di utilizzare le proprie conoscenze in diversi ambiti, quali modellare entità complesse (*User*, *VideoGame*, *DataBase*) e implementare una netta separazione delle responsabilità tra i layer applicativi: l'interfaccia utente (con *JavaFX*), la logica di business (i singoli giochi) e l'accesso ai dati (tramite JDBC e database H2).

Il progetto "SalaGiochi Virtuale" mira allo sviluppo di un'applicazione desktop in linguaggio Java che simula una sala videogiochi virtuale. Il sistema permette agli utenti di registrarsi, autenticarsi e accedere a una suite di quattro giochi classici: BattleShip, Tris, Roulette e DiceGuesser.

L'obiettivo principale è fornire una piattaforma di intrattenimento funzionale, che tenga traccia dei punteggi dei giocatori tramite un database H2 embedded persistente, offrendo una classifica globale aggiornata. I risultati attesi includono un software funzionante, privo di bug, corredato da documentazione tecnica e manuale utente.

Team di progetto: Il team è composto da 3 studenti di ingegneria informatica

Andrea Mondino: Capo Programmatore

Matilde Scioscia: Programmatore & Document Reviewer

Matteo Megna: Programmatore & Document Reviewer

2. Modello di processo

Per questo progetto è stato adottato un **modello di sviluppo iterativo e incrementale**.

Questo approccio ha permesso di rilasciare versioni funzionali intermedie. Il Team gestisce l'elicitazione dei requisiti mediante due fasi; la prima riguarda la comprensione e l'analisi dei requisiti secondo quanto richiesto a lezione, la seconda fase tratta la rielaborazione degli stessi in linguaggio naturale, affinché possano venire implementati nel progetto.

A tale proposito, il Team divide la stesura del software in 4 principali iterazioni:

1. **Core & Infrastructure:** Setup DB (H2), Gestione Utenti (Login/Register), GUI di base (Room.java). Questa fase serve al Team per tracciare le fondamenta dell'applicazione, fare chiarezza sui requisiti offerti dal sistema e iniziare a suddividere il software su tre livelli architetturali.
2. **Unified Model Language definition:** Il Team si concentra sullo sviluppo di codice partendo dai diagrammi, traducendo, in particolare dal *Class Diagram*, lo scheletro delle classi principali e il comportamento dell'applicazione.
3. **Gameplay Implementation:** tratta lo sviluppo dei 4 giochi (BattleShip, Tris, Roulette, DiceGuesser).
4. **Refinement & Finalization:** classifiche, rifinitura GUI, testing e bug fixing. Nell'ultima iterazione il focus riguarda una rifinitura generale, con un ulteriore controllo sui requisiti mancanti e sull'aggiunta di *features* che migliorino il feedback del software.

To be noted: Ogni riunione con il Team ha come primo obiettivo quello di verificare che i requisiti della fase precedente siano rispettati, correggendo eventuali mancanze e/o ritrattando i requisiti non applicabili al sistema. Inoltre sono stati applicati durante lo sviluppo del progetto alcuni software di controllo qualità del codice (*Structure Analysis*, *SonarQube*, *PMD*, *stan4J*).

3. Organizzazione del progetto

Modello Organizzativo e Metodologia: il team ha adottato un modello organizzativo ibrido, ispirato alla struttura del "**Chief Programmer Team**" (Capo Squadra Programmatore), adattato a un contesto accademico e agile.

Struttura del Team e Responsabilità: La struttura prevede una leggera gerarchia funzionale per garantire coerenza e visione d'insieme

- **Chief Programmer / Project Manager** (Andrea Mondino): ha agito come leader tecnico e coordinatore.
- **Programmatori** (Matilde Scioscia e Matteo Megna): hanno lavorato in parallelo e in stretta collaborazione con il Chief Programmer per sviluppare tutte le funzionalità principali del software.

Metodologia Collaborativa: diversamente da una gerarchia rigida, la metodologia di sviluppo si è basata su un **Pair Programming fluido**. Le coppie di programmazione non erano fisse, ma si formavano dinamicamente in base al task da svolgere, permettendo una collaborazione attiva. Questa dinamica si è manifestata in due modalità principali:

- **Pairing Leader-Membro:** il Chief Programmer è stato affiancato dagli altri membri (PP verticale) per impostare le varie funzionalità dell'applicazione (ad esempio la connessione al DB o la struttura della GUI e scrittura dei videogiochi) e per allineare la visione progettuale.
- **Pairing Membro-Membro:** i programmatori hanno lavorato in Pair Programming (orizzontale) per sviluppare alcune delle funzionalità complesse, scambiandosi i ruoli di *driver* e *navigator*.

Questa rotazione dinamica delle coppie ha garantito dei benefici chiave:

- Massima **condivisione** della conoscenza: tutti i membri hanno compreso l'intero codebase, la struttura e la logica di implementazione.
- **Revisione** continua del codice: il PP ha agito come una code review in tempo reale, aumentando la qualità del codice e riducendo i bug.

4. Norme, linee guida, procedure

Per garantire l'uniformità del codice e facilitare la collaborazione, il team ha seguito alcune norme:

- **Coding Standard:** convenzioni di denominazione standard di Java.
- **Version Control:** utilizzo di GitHub con un singolo branch principale (**main**) per costante aggiornamento reciproco.
- **Build System:** progetto configurato come **Maven Project** in Eclipse.

- **Documentazione del Codice:** Javadoc per le classi e i metodi, commenti del codice sorgente e integrazione di un file README.

5. Attività di gestione

Per la gestione del progetto il team ha seguito un approccio agile per la modellazione. Le attività principali sono state:

- **Gestione delle priorità (MoSCoW):** sono state definite delle priorità sulle funzionalità che andavano implementate man mano nella nostra applicazione
 - **Must:** DataBase persistente, navigabilità tra le stanze, account registration, suite di giochi.
 - **Should:** classifica globale, pop-up per notifiche di sistema.
 - **Could:** modalità di gioco in anonimo, conversione dei punteggi in monete virtuali.
 - **Won't:** sicurezza legata al database (nessuna *encryption* sulle password), condivisione online (*multiplayer*), hosting sulla rete dell'applicazione (funzionamento solo in locale da un eseguibile java).
- **Pianificazione delle iterazioni:** il lavoro è stato organizzato in cicli brevi (1/2 settimane), all'inizio dei quali il team si riuniva per analizzare lo stato corrente del progetto e porsi obiettivi da raggiungere entro la fine del ciclo stesso.
- **Meeting di allineamento:** brevi riunioni periodiche per verificare l'integrazione delle parti sviluppate e risolvere eventuali problemi.

6. Rischi

È stata condotta un'analisi preventiva dei rischi per identificare potenziali problemi del progetto. Considerando l'architettura del progetto basata su un *database embedded* (H2) che salva i dati su file locali, è stato identificato un unico rischio critico principale (fatta eccezione per la protezione del DataBase da attacchi SQL-injection):

- **Cancellazione o corruzione del DataBase:** poiché il database risiede su file fisici nella cartella locale del progetto (. /database/salagiochi.mv), esiste il rischio

concreto che questi file vengano cancellati accidentalmente durante le operazioni di pulizia del progetto o corrotti in caso di crash anomalo dell'applicazione. Questo comporterebbe la perdita totale degli utenti registrati e dei punteggi.

- **Ripristino Automatico e Backup:** è stata implementata la classe `DataBaseInitializer.java` che all'avvio verifica l'esistenza delle tabelle. Se il file del DB viene cancellato, il sistema è in grado di ricreare automaticamente la struttura (tabelle utente, videogame, ecc.) e popolare i dati di default, garantendo che l'applicazione non vada in crash, pur partendo da uno stato "pulito". Durante lo sviluppo, viene mantenuta una copia di backup della cartella database.

7. Personale

Il progetto ha visto il coinvolgimento attivo di 3 membri per l'intera durata dello sviluppo. L'impegno totale stimato è di circa 120 ore uomo. Avendo adottato la metodologia del *Pair Programming*, due ore uomo corrispondono ad un'ora di programmazione di coppia. Questo approccio ha privilegiato la qualità rispetto alla quantità di codice grezzo prodotto.

8. Metodi e tecniche

In questa sezione vengono dettagliati i requisiti definitivi del software e le tecniche utilizzate.

8.1 Requisiti: un'analisi più dettagliata dei requisiti funzionali e non funzionali è riportata nel documento "Specifica dei Requisiti". Di seguito vengono riassunte le macro funzionalità e i vari vincoli architetturali:

- **Gestione Utente:** Registrazione, Login, e navigazione per la selezione dei giochi.
- **Gameplay:** implementazione ed esecuzione di 4 giochi (Tris, BattleShip, Roulette, DiceGuesser).
- **Sistema di Punteggio:** aggiornamento dei punteggi e visualizzazione classifica globale.
- **Portabilità:** esecuzione su qualsiasi macchina con Eclipse e JDK 21+ (senza strumenti esterni).
- **Persistenza Dati:** database di tipo Embedded (H2).

- **Interfaccia Grafica:** librerie JavaFX.
- **Metodologia:** aderenza alla traduzione dei diagrammi UML.

Esclusioni (Out of Scope): le seguenti funzionalità sono state analizzate ma escluse esplicitamente dal perimetro del progetto attuale.

- **Applicazione multiplatforma:** il sistema non supporta l'integrazione tra macchine diverse.
- **Web Deployment:** la piattaforma non è pensata per essere inizialmente pubblicata in rete.
- **Audio e Resizing:** non sono previsti effetti sonori né interfaccia responsive.
- **Target Dispositivi Mobili:** non è previsto il supporto o l'integrazione con dispositivi mobili.
- **Eseguibile:** non viene fornito un eseguibile pre-compilato dell'applicazione.

8.2 Progettazione: l'applicazione adotta un'architettura a 3 livelli.

- **Presentation Layer (GUI):** `com.mondsciomegn.salagiochi.gui` (basato su JavaFX, gestisce l'interfaccia con l'utente).
- **Logic Layer (Game):** `com.mondsciomegn.salagiochi.videogame` (basato sulla classe astratta `VideoGames`, contiene la logica dei giochi).
- **Data Access Layer (DataBase):** `com.mondsciomegn.salagiochi.db`

8.3 Implementazione e Testing:

- **Linguaggio:** Java SE (JDK 21+).
- **GUI:** JavaFX
- **Database:** H2 Embedded SQL Database (file-based)
 - Data Access: JDBC nativo, senza API.
- **Testing:** la strategia di verifica combina test automatici e manuali
 - Test Automatici (JUnit): sviluppati per verificare la correttezza della logica di business (es. validazione credenziali Login) e il calcolo degli score dei videogiochi.
 - Test Manuali: eseguiti per validare l'interfaccia grafica (GUI) e migliorare l'esperienza utente.

9. Garanzia della qualità

La qualità è stata garantita tramite:

- **Test Automatici (JUnit):** è stata implementata una suite di test automatici utilizzando il framework JUnit. Questo ha permesso di verificare in modo deterministico le funzionalità critiche (come l'autenticazione e l'assegnazione dei punteggi da parte dei videogiochi).
- **Pair Programming:** questa metodologia ha fornito una revisione continua e completa del codice.
- **Small Releases:** i rilasci frequenti e periodici assicurano che i vari requisiti siano correttamente implementati.
- **Java Conventions:** il team ha rispettato rigorosamente le convenzioni standard di Java; questo approccio ha garantito la produzione di un codice uniforme e più leggibile.
- **Reusability:** parti di codice comune vengono riutilizzate e non riscritte, ciò rende la manutenzione più veloce e funzionale.

10. Organizzazione dei pacchetti di lavoro

Il progetto è stato suddiviso nei seguenti Work Packages (WP):

1. **Stesura Project Plan e sviluppo diagrammi UML.**
2. **Setup: configurazione ambiente Maven/GitHub.**
3. **Architettura & DataBase (Package `.db`):** implementazione della struttura relativa al DataBase e creazione delle classi-entità (User, Category, ...).
4. **GUI e Autenticazione (Package `.gui`):** sviluppo della classe `Room.java`, logica di Login/Register, gestione degli account e modalità di gioco anonima.
5. **Sviluppo Giochi (Package `.videogame`):**
 - a. Creazione classe astratta `VideoGames`
 - b. Implementazione `Tris.java`

- c. Implementazione BattleShip.java
 - d. Implementazione Roulette.java
 - e. Implementazione DiceGuesser.java
6. **Classifiche & Integrazione:** integrazione delle ultime funzionalità del database, come Tabellone e Score.
7. **Testing & Debugging:**
- a. Test manuali svolti per la maggior parte sulla grafica dell'applicazione (GUI).
 - b. Test automatici con JUnit per il controllo credenziali di Login e per il controllo degli score di tutti i videogiochi.

11. Risorse da utilizzare

- **Hardware:** laptop con qualsiasi OS.
- **IDE:** Eclipse IDE for Java Developers + JDK 21+.
- **Build Tool:** Maven.
- **Language:** Java SE
- **GUI Libraries:** JavaFX.
- **DataBase:** H2 Embedded Database.
- **Sharing Tools:** GitHub, Google Drive.

12. Budget e programma

- **Budget:** il progetto ha costo monetario zero.
- **Programma di massima:**
 1. **Analisi e Avvio:** stesura Project Plan, diagrammi UML e setup GitHub/Maven.
 2. **Scheletro del progetto:** sviluppo architettura e DataBase, GUI e logica di Login/Register, navigabilità tra le stanze.
 3. **Implementazione gameplay:** sviluppo e integrazione dei 4 giochi.
 4. **Integrazioni finali e consegna:** creazione del Tabellone per la classifica degli utenti, esecuzione test JUnit e rilascio finale.

13. Cambiamenti

- **Gestione delle modifiche:** qualsiasi modifica ai requisiti iniziali è stata discussa dal team nei vari incontri e nelle sessioni di Pair Programming.
- **Adattabilità del software:** l'architettura del software è stata progettata per minimizzare l'impatto di eventuali cambiamenti futuri. L'aggiunta di nuovi giochi richiederebbe un cambiamento minimo della classe `Room.java` senza ulteriori modifiche al database.

14. Consegna

La consegna finale avverrà secondo le modalità richieste, includendo:

- **Folder con il codice sorgente:** repository GitHub completa.
 - **DataBase:** cartella database/ contenente i file `salagiochi.mv.db` e `salagiochi.trace.db`.
- **Folder con i documenti:** Project Plan e documentazione correlata, Diagrammi UML.