

Project Plan

Versione	Data	Descrizione
1.0	17 Novembre 2025	Prima versione ufficiale del documento Project Plan
1.1	19 Gennaio 2026	6) Modifica del plugin per l'implementazione dell'interfaccia grafica 4) Aggiunta procedura per il testing del codice 7) Piccole modifiche alle responsabilità del team 8) Aggiunti metodi e tecniche per l'implementazione (Maven, Papyrus, JavaFX, plugin per la qualità del codice), per il testing (JUnit). Aggiunto modello MoSCoW per prioritizzare i requisiti e una scala numerica per stimare il carico di lavoro per il suo sviluppo.

1.Introduzione

Il nostro team di sviluppo nasce dalla necessità di un casinò di offrire un servizio specializzato nel Poker online multigiocatore per:

- aumentare i propri ricavi tramite la popolarità acquisita grazie allo sviluppo del servizio online
- offrire un servizio gratuito con il quale gli utenti possono sperimentare la loro fortuna nel gioco del poker senza rischi economici

I responsabili del progetto sono dunque:

- Il casinò, cliente e diretto interessato al prodotto finale
- Il team di sviluppo "THR-Project", composto da
 - Alessandro Cardillo Ottaviano
 - Marian Stefan Dolete
 - Oleh Kalkovets Cestonaro
 - Leonardo Forchini

2.Modello di processo

Utilizzeremo il framework agile **Scrum**, che utilizza lo sviluppo incrementale del software seguendo questi passaggi:

- **Product Backlog:** il product owner si occuperà di mantenerlo aggiornato
 - nuove funzionalità (da implementare con requisiti funzionali)
 - requisiti non funzionali
 - bug e problemi
- **Sprint Planning:** il team di sviluppo e il product owner collaborano per estrapolare gli elementi più importanti dal product backlog, stabilendo gli obiettivi per la sprint
- **Sprint:** periodo fisso di tempo durante il quale la squadra lavora per raggiungere gli obiettivi fissati nella fase precedente, garantendo quindi che alla fine di ogni sprint il software sia funzionante
- **Daily Scrum:** ogni giorno durante una sprint la squadra condivide informazioni sul progresso, identifica ostacoli e pianifica le attività per il giorno successivo

- **Sprint Review:** alla fine di ogni sprint la squadra presenta il lavoro al cliente che potrà fornire feedback inerenti al software. Il product owner verifica assieme al cliente se il lavoro è conforme alle aspettative e valuta eventuali modifiche al backlog per rispondere ai feedback
- **Sprint Retrospective:** la squadra condivide informazioni utili al miglioramento del processo di sviluppo (non per migliorare il software)

In parallelo a queste fasi utilizzeremo una **Kanban Board** per monitorare lo stato delle attività e il carico di lavoro istantaneo.

3. Organizzazione del processo

Il progetto è gestito secondo la metodologia agile **Scrum**, quindi si distinguono tre ruoli:

- **Scrum Master:** facilita e garantisce il processo Scrum gestendo gli impedimenti e promuovendo il miglioramento del team
- **Product owner:** rappresenta il cliente, quindi è responsabile della visione del prodotto e delle priorità (gestisce il product backlog)
- **Development team (team di sviluppo):** team autonomo che pianifica le attività delle sprint e sviluppa, testa e integra il software

4. Standard, linee guida e procedure

Standard di sviluppo:

- Codice: rispettare le convenzioni java per la scrittura del codice
- Versionamento: utilizzare la funzione branch di GitHub
- Qualità: testare con test unitari e di integrazione ogni incremento

Linee guida operative:

- Task: gestione e visione con kanban board
- Riunioni scrum: obbligatorie e documentate

Procedure:

- Merge: creare diversi branch per l'implementazione di nuove funzionalità e testarli prima del merge con il main
- Bug e problemi: aprire issue su GitHub per la gestione di bug e problemi nel software
- Test: per il controllo del codice prima del merge con il branch main e ad ogni incremento del branch in sviluppo

5. Attività di gestione

Seguiremo queste fasi (in ordine cronologico) in modo iterativo:

- Product backlog e Sprint planning: pianificati prima di ogni Sprint
- Sprint: gli assegneremo una durata settimanale
- Daily Scrum: li simuleremo con una cadenza di ogni 2/3 giorni
- Sprint review e retrospective: pianificati alla fine di ogni Sprint

6. Rischi potenziali

- Il team non conosce a pieno il framework per lo sviluppo dell'interfaccia grafica del software e i metodi per l'implementazione della modalità multiplayer online
- Sottostima delle attività che potrebbe portare all'aggiunta di requisiti e quindi all'aumento del carico di lavoro
- L'impossibilità di fare Daily Scrum tutti i giorni porterà ad un aumento dei tempi di sviluppo

7. Personale

RUOLO	PERSONALE	RESPONSABILITÀ
Scrum Master	Alessandro Cardillo Ottaviano	Facilita e garantisce il processo Scrum promuovendo il miglioramento del team.
Product Owner	Alessandro Cardillo Ottaviano Dolet Marian Stefan Leonardo Forchini Oleh Kalkovets Cestonaro	Simulare i bisogni e le aspettative del cliente compilando il product backlog e gestendo le priorità dei requisiti. Aggiornare questo documento in base allo stato dello sviluppo.
Team di sviluppo	Alessandro Cardillo Ottaviano Dolet Marian Stefan Leonardo Forchini Oleh Kalkovets Cestonaro	Redigere la documentazione necessaria. Sviluppare e testare il codice. Modellare con UML.

8. Metodi e tecniche

Ingegneria dei requisiti

- Alcuni dei requisiti funzionali verranno estrapolati dal regolamento ufficiale del gioco del Poker, altri verranno simulati dal product owner
- I requisiti verranno sottoposti a criteri di accettazioni per verificarne la validità
- Ad ogni requisito verrà assegnata una priorità secondo il modello MoSCoW
 - M: Must have, indispensabile
 - S: Should have, importante ma non critico
 - C: Could have, migliorativo
 - W: Won't have, escluso per il momento
- Ad ogni requisito verrà assegnato un peso in termini di carico di lavoro

Progettazione

- La progettazione sarà incrementale e verrà documentata di volta in volta
- Utilizzeremo modelli UML per modellare l'architettura e il funzionamento

Implementazione

- Lo sviluppo sarà incrementale, in modo che alla fine di ogni Sprint il codice sia funzionante, per poterlo mostrare al cliente
- La gestione della configurazione del software verrà gestita con GitHub (che si occuperà anche del versionamento)
- Utilizzeremo Maven per la gestione delle dipendenze del progetto
- L'implementazione dell'interfaccia grafica avverrà grazie al framework JavaFX
- Utilizzeremo Papyrus per lo sviluppo del class diagrammi UML e per la generazione automatica del codice
- Per creare gli altri UML utilizzeremo LucidChart
- Per garantire qualità del codice utilizzeremo plugin come Stan4J e EclipsePMD

Test

- Le apparecchiature di prova necessarie non saranno altro che le macchine personali degli sviluppatori. La stessa cosa sarà applicata per quanto riguarda l'ambiente di prova.
- I test verranno eseguiti mediante la compilazione del codice in ambiente Eclipse IDE, si partirà dall'implementazione per poi effettuare il successivo collaudo di tutta la parte che riguarda i meccanismi "back-end" del software e quindi successivamente anche l'interfaccia utente.
- L'utilizzo del framework JUnit aiuterà nella creazione/utilizzo di test sui casi d'uso del dominio del progetto.

9. Garanzia di qualità

La garanzia di qualità deve assicurare che il software sviluppato soddisfi i requisiti funzionali, non funzionali e gli standard concordati:

- il product owner si occuperà di definire i requisiti che determinano la qualità del prodotto;
- il team di sviluppo garantirà la qualità del codice rispettando le convenzioni java ed eseguendo i test;
- lo scrum master faciliterà il processo.

10. Work packages

I work packages corrisponderanno alla suddivisione del progetto in macro-aree formate in base alle funzionalità comuni dei loro componenti (ciò porterà ad un utilizzo più chiaro ed efficiente dei file di codice e sarà più semplice suddividere le attività nel team):

- Grafica;
- Multiplayer client/server;
- Gioco;
- Database.

11. Risorse

- Hardware

- ogni membro del THR-Project ha a disposizione un calcolatore;
 - il server sarà in cloud;
 - il database sarà embedded;
- Software
 - Server in cloud
 - GitHub
 - IDE Eclipse
 - Papyrus
 - Maven
 - JavaFX
- Umane
 - THR-Project team;

12. Budget e programma

- Progetto universitario senza scopo di lucro;
- La programmazione e la gestione delle attività avverrà tramite Sprint planning e Daily Scrum;

13. Cambiamenti

Scrum è un metodo agile, quindi è orientato al cambiamento:

- le modifiche ai requisiti: gestite attraverso la modifica del product backlog da parte del product owner, che le valuta e le assegna una priorità
- le modifiche al codice e alla documentazione: gestite tramite GitHub

14. Consegna

La consegna del progetto avverrà pubblicando il tutto in un repository github da condividere preventivamente con il professore (account github: garganti) e con l'esercitatrice (account github: silviabonfanti). Il progetto deve essere completato 5 giorni prima della sua presentazione e dovrà contenere:

- tutta la documentazione necessaria;
- tutto il codice funzionante;

Il prodotto finale dovrà essere importabile ed eseguibile sull'IDE Eclipse.