

**Progetto di ingegneria del software, anno accademico 2022-2023**

**Anno accademico 2022-2023**

**Membri del team**

* Lorenzo Magni, matricola 1073257
* Saif Bouchemal, matricola 1074800
* Marianna Romelli, matricola 1072382

**Sommario**

**Il nostro progetto**

**Project plan**

Introduzione

Modello di processo

Organizzazione del progetto

Standards, linee guida e procedure

Attività di gestione

Rischi

Membri

Metodi e tecniche

Garanzie di qualità

Package di lavoro

Risorse

Budject e pianificazione

Cambiamenti

Consegna

Software Life Cycle

Configuration Management

Struttura del progetto

Issue

Branch

People Management

Software quality

Parametri riguardanti l’operatività del software

Parametri riguardanti la revisione del software

Parametri riguardanti la transizione verso un nuovo ambiente

**Il nostro progetto**

Si crei un gioco online disponibile su desktop chiamato Lupus in fabula.

**I personaggi**

Narratore, Cittadini, Lupi Mannari, Veggente, medium (da 10 giocatori in su), guardia del corpo (da 12 giocatori in su), indemoniato (da 15 giocatori in su), massoni (da 15 giocatori in su), gufo (da 15 giocatori in su), criceto mannaro (da 15 giocatori in su), mitomane (da 15 giocatori in su)

**I ruoli**

Narratore: chiama il giorno e la notte e durante la notte chiama i ruoli e dirige

Cittadini: durante il giorno decidono chi linciare e di notte dormono

Lupi mannari: di solito sono 2 se in 15+ diventano 3, di notte decidono chi uccidere

Veggente: di giorno è un cittadino normale ma di notte può scegliere qualcuno e vedere se è un lupo

Medium: di notte può vedere se un morto è o no un lupo

Guardia del corpo: dalla seconda notte può indicare qualcuno e proteggerlo dai lupi

Indemoniato : è un cittadino comune ma patteggia per i lupi

Massoni: SOLO la prima notte si riconosceranno e si difenderanno a vicenda

Gufo: ogni notte indicherà una persona e la manda allo spareggio se si gioca in 20 o più sarà direttamente uccisa

Criceto mannaro: di notte non può essere ucciso dai lupi ma se la veggente ne scopre il ruolo lo uccide

Mitomane: dalla seconda notte può indicare qualcuno e se becca un lupo o un veggente ne diventa una copia

**Lo svolgimento**

Il gioco prevede 2 fasi che si alternano, il giorno e la notte. durante il giorno i cittadini discutono su chi dovranno linciare, di notte il narratore chiamerà tutti i ruoli e avranno un tempo prestabilito per svolgere i propri ruoli.

i *Cittadini* vincono se eliminano i lupi, i *Lupi* vincono se arrivano in numero pari ai cittadini, l’*Indemoniato* vince se vincono i lupi, il *Criceto Mannaro* vince se arriva vivo alla fine.

N.B. i lupi decidono insieme chi uccidere, i morti diventano fantasmi diventano cittadini votando ma senza poter parlare, il narratore di notte chiama sempre tutti i ruoli (tranne i massoni dalla 2° notte)

**Project Plan**

Sviluppiamo l’applicativo per una community immaginaria. Inizialmente i requisiti li decidiamo noi e nel momento in cui l’applicazione potrà essere disponibile per gli utenti saranno loro a fare proposte in base a necessità e funzioni che vorrebbero fossero implementate.

La nostra azienda è una società di sviluppo software costituita da Lorenzo Magni, Saif Bouchemal e Marianna Romelli al fine di progettare l’intera applicazione iniziale, in modo che possa poi essere il prototipo per possibili migliorie richieste dagli utenti o finanziatori.

Il fine del nostro lavoro è di mettere sul mercato un prototipo del gioco per possibili futuri investitori per ricevere un finanziamento futuro.

Per quanto riguarda il compenso salariale, inizialmente non ci sarà e il nostro lavoro sarà finanziato successivamente dal possibile investitore. Saremo disposti a prendere in considerazione requisiti aggiuntivi richiesti dagli investitori se necessario al fine di migliorare o modificare l’applicazione come richiesto.

La community sarà formata dagli utenti che hanno avuto l’occasione di visionare l’applicazione e che possono dare un contributo significativo ad essa.

1. **Introduzione**

Dopo un’attenta analisi dei requisiti scritti da noi mettendoci nei panni di un possibile utente, di seguito saranno riportati i passo al fine di realizzare una prima versione del sistema richiesto.

Alla progettazione del sistema e allo sviluppo del primo prototipo lavorerà tutto il team composto da Lorenzo, Saif e Marianna. Successivamente saranno precisati i ruoli specifici per ognuno.

Dopo un primo incontro si decide di sviluppare fisicamente un’applicazione che implementi tale funzionalità:

* Sviluppo del gioco nella versione base funzionante
* Inserimento della lingua inglese oltre alla lingua italiana
* Networking
* Interfaccia grafica
* Inserimento della chat testuale

Abbiamo stabilito che la prima versione sia disponibile per la fine di gennaio 2023, quindi abbiamo all’incirca 70 giorni.

Ci saranno degli incontri settimanali durante la fase di progettazione del sistema in modo da collaborare e discutere insieme nonché avere un feedback dell’intero gruppo di lavoro.

Entro metà dicembre metteremo il prototipo online in modo che gli utenti possano iniziare ad utilizzarlo ed a mandarci alcuni requisiti che vorrebbero implementare.

1. **Modello di processo**

Per il processo di sviluppo del prototipo abbiamo optato per un approccio di tipo Agile, infatti puntiamo a rilasciare una versione di base del prodotto e integrarla mano a mano con piccoli aggiornamenti che porteranno con sé le funzionalità descritte sopra, in modo da facilitare anche le fasi di testing e debugging.

Fondamentalmente sarà inizialmente le nostre idee che faranno si che il processo di sviluppo venga migliorato e implementato e successivamente saranno invece gli utenti della community.

Per tenere traccia dello sviluppo delle funzionalità richieste, optiamo per l'utilizzo delle issue su Github, che saranno assegnate settimanalmente a ogni componente del gruppo in base a quello che si decide nei meeting. Mano a mano che ognuno andrà a completare i punti che gli sono stati assegnati, le issue verranno chiuse e i documenti pubblicati sulla piattaforma.

1. **Organizzazione del progetto**

Le persone coinvolte nel progetto sono inizialmente solo il tema di progettisti/sviluppatori e in seguito anche gli utenti della community.

L'intera documentazione riguardante il progetto sarà riportata e tracciata su GitHub. In particolare, il team si incontra settimanalmente per analizzare il lavoro svolto dall'ultimo meet e pianificare le attività future, tracciando i task come issue su GitHub in modo che ogni persona del team sia al corrente di cosa stiano facendo gli altri.

1. **Standards, linee guida e procedure**

Il team ha deciso di sviluppare un'applicazione per implementare il sistema descritto. L'applicazione sarà suddivisa in due parti:

* ○  *Client*: si occupa dell'interfaccia grafica e di interagire con il server;
* ○  *Server*: gestisce l’andamento della partita/gioco.

Il grafica del Client verrà sviluppato utilizzando OpenGL.

1. **Attività di gestione**

Abbiamo deciso di organizzarci secondo la filosofia dello "swat team" (Skilled Workers with Avdanced Tools): gli incontri saranno settimanali e relativamente brevi, della durata di un'ora circa. Durante gli stessi potranno essere effettuate sessioni di brainstorming per condividere idee e risolvere problemi.

Il team di sviluppo si pone l'obiettivo di incontrarsi settimanalmente per aggiornarsi sullo stato dei lavori. In base alle esigenze si decide se effettuare la riunione virtualmente o se è necessario incontrarsi e svolgerla di persona.

Ad ogni incontro, i membri del gruppo espongono brevemente quali sono stati i punti portati avanti e quali sono state le problematiche incontrate sul proprio cammino. Mano a mano che si procede nell'esposizione, si aggiorna un documento che tiene traccia dei lavori svolti fino a quel momento. Alla fine di ogni meeting verrà scritta una breve relazione in merito a quanto discusso.

Alla fine del meeting ci si accorda su quali sono i punti da portare avanti; sarà il project manager ad assegnare a ciascun membro del gruppo, se stesso compreso, i compiti che dovranno essere eseguiti prima del prossimo incontro.

1. **Rischi**

La durata è di 70 giorni e l’obbiettivo è quello di arrivare al termine con tutta la documentazione e il prototipo definito. In particolare abbiamo stabilito come scadenza del 15 dicembre per la consegna dell’applicazione con le prime funzionalità così che la community possa usufruirne ed il 20 gennaio per l’applicazione implementata con le modifiche effettuate successivamente dalle richieste degli utenti.

Essendo questo progetto inizialmente non retribuito abbiamo deciso di utilizzare dei finanziamenti di altri fondi per tale progetto.

1. **Membri**

Utilizzando una modalità di sviluppo Agile, non c'è una vera e propria distinzione fra i ruoli nel team. Inoltre, i task, decisi in comune accordo con il project manager, verranno inseriti come issue su Github e designati ad un programmatore. Nonostante questo, in una panoramica più generale possiamo definire i seguenti ruoli:

○  Project Manager:

○  Progettista Software:

○  Progettista Database:

○  Back-end Developer:

○  Front-end Developer:

○  Tester:

1. **Metodi e tecniche**
2. **Garanzie di qualità**

Il team punta a sviluppare un software che rispetti i parametri di qualità indicati dal modello di McCall:

○ Correttezza

○ Affidabilità

○ Robustezza

○ Integrità

○ Usabilità

Per garantire la sicurezza utilizziamo librerie moderne, aggiornate frequentemente e

poco inclini ad avere vulnerabilità.

Inoltre dividiamo il lato client da quello server, cosicché in futuro il cliente potrà assumere personale separato per lo sviluppo delle due parti.

Puntiamo a sviluppare un software che, oltre a rispettare i quality assurance, sia anche sicuro.

1. **Package di lavoro**

Il protocollo di lavoro sarà costituito da 3 parti principali: il lato client, il lato server e una parte comune. In questo modo la parte relativa all’interfaccia grafica sarà presente del codice client, la parte di gestione dei dati sul lato server e le parti in comune saranno condivise con il lato client e server.

1. **Risorse**

Le risorse designate al progetto sono un project-manager e due sviluppatori che lo affiancano. Il team utilizza la IntelliJ per la scrittura di codice e per condividere con tutto il team il lavoro svolto si utilizza GitHub. Anche per quanto riguarda le issue usiamo Github in modo da avere tutto il lavoro passo a passo insieme.

Al team vengono forniti 3 computer, una connessione internet, una stanza ed un ufficio per collaborare. Il team può decidere di incontrarsi di persone per discutere del progetto e per collaborare in ufficio oppure si può decidere tranquillamente di svolgere il proprio lavoro da casa collaborando attraverso Discord.

1. **Budject e pianificazione**

Al momento Lorenzo è a 10,5 ore, Marianna a 8,5 Saif 2

1. **Cambiamenti**

Oltre alle feature base incluse inizialmente nel prodotto e descritte al punto uno di questo documento, l'obiettivo sarà quello di mettere le basi per ulteriori funzionalità allo scopo di fornire un servizio più completo, in accordo con la community. Queste verranno inserite nelle successive versioni del software e includeranno:

* Altre lingue oltre all’italiano e all’inglese
* Guida del gioco per una comprensione migliore
* Crittografia chat testuale
* Chat vocale

1. **Consegna**

Gli incontri sono fissati ogni settimana, il giorno può variare in base agli incontri ma abbiamo comunque stabilito una frequenza settimanale in modo da rimanere aggiornati con lo sviluppo e concordare insieme i prossimi passi da svolgere. Per quanto riguarda la prima settimana di lavoro abbiamo deciso di trovarci 3 volte per concordare come procedere e come suddividere il lavoro. In particolare ci siamo trovati due volte di persona e una volta online.

**Software Life Cycle**

Per il processo di sviluppo il team ha scelto un approccio di tipo Agile poiché meglio si adatta alla nostra metodologia di lavoro:

* Consideriamo importante il team, le abilità e le interazioni tra di noi. I lavori vengono assegnati in base alle capacità che ognuno possiede. Se uno della squadra si accorge che il compito va oltre le sue competenze, avvisa gli altri e se ne discute insieme.
* Nel team non c’è una struttura di tipo gerarchico: ci consideriamo tutti allo stesso livello. Questo ci porta ad essere più compatti e motivati.
* Consideriamo più importante un prodotto software funzionante che spendere troppo tempo sui documenti: a sviluppo iniziato, se ci accorgiamo che ci sono punti da implementare che non avevamo in precedenza considerato, li andiamo ad aggiungere poi alla documentazione.
* Utilizziamo la tecnica del pair programming: per lo sviluppo dei lati front-end e back-end, gli sviluppatori hanno l’opportunità di lavorare in coppia: uno dei due scrive il codice mentre l’altro funge da supervisore e revisore; i due possono scambiarsi di ruolo.
* Durante lo sviluppo del software utilizziamo la tecnica del timeboxing: suddividiamo lo sviluppo del codice in intervalli temporali entro i quali dobbiamo sviluppare determinate funzionalità.
* Diamo molta importanza alla collaborazione con gli utenti, infatti dove aver messo appunto le funzionalità base dell’applicazione mettiamo subito a disposizione il gioco in modo che gli utenti della community possano darci consigli su come migliorare le funzionalità. Gli utenti non collaborano con noi dal punto di vista organizzativo, ma ci aiutano nell’individuare alcune funzioni aggiuntive che vorrebbero fossero implementate.
* Il team è propenso al cambiamento e punta a rispondegli in maniera adeguata. Quando gli utenti avanzano delle richieste, cerchiamo di fare in modo di accontentare tutti stanto in ogni modo attenti a quello che si è sviluppato fino a quel momento.
* In caso che il nostro prodotto venga richiesto da possibili investitori saremo lieti di prendere in considerazioni le proposte. Se l’investitore vorrà poi avere delle funzionalità aggiuntive saremo lieti di prenderle in considerazione. Prima di un possibile finanziamento faremo provare l’applicazione anche al finanziatore.

Come approccio di progettazione software per lo sviluppo del sistema ci siamo attenuti alla model-driven architecture (MDA), o architettura guidata dal modello.

**Configuration menagement**

Tutto il lavoro svolto che si tratti di documentazione o dic odice viene salvato in una repository su Github in condione con tutti i membri del team.

Struttura del progetto

All’interno della repository sono presenti 2 cartelle:

* La prima cartella chiamata code contiene all’interno tutto il codice del progetto. Le sottocartelle sono 3 più il file pom. Nella prima sottocartella abbiamo il codice appartentente al client, la seconda è in comune con client e server e la terza è appunto il server. la cartella common è stata importata come libreria in server e client.
* La seconda chiamata docs contiene tutta la documentazione a riguardo. Esso conterrà tutta la documentazione effettuata e le linee guida per lo sviluppo del gioco, nonché la spiegazione del gioco.

Issue

Nei vari meeting settimanali vengono creati i macro task che dovranno essere sviluppati nel corso della settimana. Le varie attività sono create come issue su GitHub, aggiungendo una breve descrizione del lavoro e la persona di riferimento che si occuperà di portarla a termine. Durante la settimana ogni membro del team è libero di creare ulteriori issue se lo ritiene necessario.

Le issue possono trovarsi in vari stati a seconda del loro avanzamento. Per tenere traccia dello stato si utilizza una classica board Kanban suddivisa in 5 colonne che sono To Do, Progress, Pull Request, Testing e Done:

* *To do*: contiene le issue che sono state create e che devono essere sviluppate.
* *Progress*: quando il responsabile della issue inizia a lavorarci, la sposta in questa colonna. In questo modo gli altri membri del team e il cliente stesso sanno a che punto si trova lo sviluppo di una determinata issue.
* *Pull Request*: quando il responsabile della issue apre una pull request per poter unire il proprio lavoro al branch master, la issue si trova in questo stato.
* *Testing*: una volta approvata la pull request, se è stato sviluppato del codice, l’issue passa in questo stato, altrimenti passa direttamente allo stato Done. Il nostro tester si occuperà di verificare che il codice sviluppato soddisfi i requisiti descritti all’interno del task. Se tutto funziona correttamente allora passa allo stato successivo (Done), altrimenti lo stato torna in To Do con un commento contenente una motivazione per tale scelta.
* *Done*: le issue che si trovano in questo stato sono state chiuse e riviste da almeno un’altra persona.

Branch

Il branch principale è il main. Tutta la documentazione si trova in versione stabile, mentre è il branch dev che contiene il codice in fase di sviluppo.

Utilizziamo sempre il branch dev finché non si arriva ad una versione stabile del codice e si arriva dunque a una versione funzionante.

**People menagement**

L’azienda ha deciso di basarsi si una struttura organizzativa non gerarchica. Utilizzando questa, abbiamo una grande capacità di adattamento e non abbiamo una vera e propria divisione prefissata del lavoro; è infatti il team a decidere come dividere e assegnare i vari task. Inoltre, in caso di difficoltà durante lo sviluppo, due membri del team possono affiancarsi per risolvere il problema in coppia. Il vantaggio di utilizzare l'adhocrazia è che abbiamo un'organizzazione flessibile, adattabile e informale che è definita da una mancanza di struttura formale.

Il team è organizzato secondo il modello SWAT (skilled worker with advanced tools). Abbiamo quindi una squadra composta da tre persone che sviluppano utilizzando canali di comunicazione informali (issue su github). Le riunioni tra il team sono tutte informali, mentre abbiamo deciso di fissare con una frequenza settimanale una riunione per condividere idee e passi come procedere.

A nostro parere, il vantaggio di rifarsi ad un’architettura non gerarchica e ad un modello SWAT è che ogni comeponenete del team sviluppa software e implementa funzionalità di componenti che lo affascinano maggiormente. Vogliamo quindi non solo arrivare all'obiettivo finale di sviluppare software e svilupparlo bene, ma di costruirlo anche con piacere.

Anche se non abbiamo una vera e propria distinzione del lavoro, possiamo riassumere in maniera più generale i settori di sviluppo e le persone che ne dedicano più tempo:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Project manager | Progettista software | Progettista database | Lato Client | Lato Server | Testing |
| Lorenzo |  |  |  |  |  |  |
| Saif |  |  |  |  |  |  |
| Marianna |  |  |  |  |  |  |

**Software Quality**

Il team si è prefissato l’obiettivo di sviluppare un’applicazione che rispetti i parametri e gli attributi di qualità definiti da McCall-Richards-Walters nel documento da loro redatto nel 1977, di seguito elencati e suddivisi in categorie come indicato dagli autori.

Parametri riguardanti l’operatività del software

* *Correttezza* – L’applicazione da noi realizzato soddisfa i requisiti e le specifiche individuate all’inizio del lavoro. Le funzionalità da noi non sviluppate ma solo progettate saranno portate avanti dal team di sviluppo del futuro investitore che assumerà dopo che avremo consegnato il prototipo.
* *Affidabilità* - L’applicazione è affidabile in quanto revisionato a livello di team e sottoposto a una lunga fase di testing prima del rilascio.
* *Efficienza* - Le risorse utilizzate dal prodotto sono limitate in quanto si tratta di un’applicazione web e molte delle operazioni sono effettuate lato server. È quindi comunque necessaria una connessione internet e un browser per accedere ai servizi del software da noi sviluppato.
* *Integrità* - Il prodotto che sviluppiamo è sicuro:
  + *I messaggi in chat sono criptati e nessun dato degli utenti è salvato sul server*.
  + *Non sfruttiamo servizi esterni*
* *Usabilità* - Il prodotto è semplice da utilizzare, infatti non sono richieste particolari abilità per poterne usufruire. L’uso dei servizi è facilitato su tutti i tipi di piattaforma grazie all’utilizzo delle librerie che permette di avere componenti grafici semplici e facili da utilizzare.

I requisiti “base” includono:

* + Avere una connessione internet
  + Sapere come si gioca a Lupus in fabula ed attraverso la guida inserita della versione successiva dell’applicazione tale requisito non è più richiesto
  + Avere un pc funzionante (Linux, Mac o Windows)

Parametri riguardanti la revisione del software

* *Manutenibilità* - La fase di individuazione degli errori è semplificata dalla nostra scelta di separare i lati server e client. Il processo è inoltre facilitato poiché si ha la parte grafica separata dalla parte di gestione dei dati: è semplice fare manutenzione avendo persone specializzate in ambiti differenti.
* *Testabilità* - Tutte le feature incluse nel prototipo sviluppato sono testabili (tramite test manuali e test automatici) prima del rilascio dell’applicazione al pubblico. Delle funzionalità che saranno invece da noi solo progettate si occuperanno in futuro i dipendenti della società dell’investitore: saranno loro a svilupparle e a effettuare la fase di testing.
* *Flessibilità* - Come per la manutenibilità, anche il processo di modifica, adattamento e perfezionamento dell’applicazione è facilitato dalla scelta di separare i lati client e server.

Parametri riguardanti la transizione verso un nuovo ambiente

* *Portabilità* - È possibile utilizzare i servizi offerti dal nostro prodotto da qualsiasi dispositivo (Linux, Mac o Windows) dotato di una connessione internet. Il server opera su qualsiasi dispositivo senza limiti, mentre il client ha bisogno di scheda video (anche integrata). Sfruttiamo delle librerie native per i vari sistemi operativi che sono già integrati e disponibili nel programma.
* *Riusabilità* - Per la parte grafica utilizziamo OpenGL.
* *Interoperabilità* - Nel futuro la nostra applicazione potrà essere integrato con altri servizi in quanto sviluppiamo in modo separato client e server.