

Rapporto del Team 3

Progetto di Ingegneria del Software

Enis Brajevic, Mattia Ferrarini, Matteo Fornaini,
Lorenzo Giarrusso, Enrico Mazzotti, Cristian Orsi

18 dicembre 2024

Indice

1 Descrizione del prodotto	4
1.1 Scope	4
1.2 Backlog	4
1.3 Diagramma dei casi d'uso	6
1.4 Diagramma dell'architettura	7
1.5 Mockup	8
1.6 Analisi dei rischi	8
1.6.1 Rischi tecnici	8
1.6.2 Rischi operativi	9
1.6.3 Rischi legati all'infrastruttura	10
1.6.4 Rischi di progetto	10
1.6.5 Tabella dei rischi	11
1.6.6 Conclusioni	11
1.7 Analisi della fattibilità	11
1.7.1 Disponibilità delle risorse	11
1.7.2 Fattibilità economica	12
1.7.3 Fattibilità commerciale	12
1.7.4 Fattibilità tecnica	12
1.7.5 Fattibilità legale	12
1.7.6 Fattibilità temporale	13
1.7.7 Conclusioni	13
2 Descrizione degli sprint	14
2.1 Note generali	14
2.2 Sprint 1	14
2.2.1 Spint Goal	14
2.2.2 Sprint Backlog	14
2.2.3 Incremento	15
2.2.4 Definition of Done	15

2.2.5	Esempi di test per ogni US	15
2.2.6	Burndown chart	16
2.2.7	Retrospettiva con Essence	17
2.3	Sprint 2	18
2.3.1	Spint Goal	18
2.3.2	Sprint Backlog	18
2.3.3	Incremento	18
2.3.4	Definition of Done	19
2.3.5	Esempi di test per ogni US	19
2.3.6	Burndown chart	20
2.3.7	Retrospettiva con Essence	20
2.4	Spint 3	21
2.4.1	Spint Goal	21
2.4.2	Sprint Backlog	21
2.4.3	Incremento	22
2.4.4	Definition of Done	22
2.4.5	Esempi di test per ogni US	22
2.4.6	Burndown chart	23
2.4.7	Retrospettiva con Essence	24
2.5	Andamento tabelle autovalutazione	25
3	Descrizione del processo	27
3.1	Autodescrizione del team	27
3.2	Risultato del team-building	27
3.3	Sintesi dei dati	28
3.3.1	Dati di gitinspector	28
3.3.2	Distribuzione temporale dei commit	30
3.4	Descrizione dell'utilizzo degli strumenti di comunicazione	32
3.5	Descrizione dell'utilizzo di LLM	33
3.6	Schermata finale di SonarQube sul codice prodotto	33
3.7	Diagramma di deployment del prodotto	35
3.8	Continuous integration e continuous development	36
3.9	Retrospettiva finale con Essence	37
3.10	Pratiche agili utilizzate	40
4	Demo	41
4.1	Link alla demo	41

5	Lista e descrizione degli artefatti	42
5.1	Product Backlog	42
5.2	Sprint Backlog	43
5.3	Incremento	44
5.4	Burndown Chart	44
5.5	Definition of Done (DoD)	44
6	Prodotto live	45
6.1	Prodotto live	45
A	Prompt e risposte degli LLM	46
A.1	Supporto alla creazione del Product Backlog	46
A.2	Supporto all'analisi della fattibilità	49
A.3	Supporto alla scrittura di test	52
A.3.1	Esempio con ChatGPT	52
A.3.2	Esempio con Copilot	53
A.4	Supporto all'inizializzazione di nuove funzionalità	55

Capitolo 1

Descrizione del prodotto

1.1 Scope

Social Backgammon è una web application che permette di giocare online a backgammon tra due o più persone. Come il nome suggerisce, il prodotto si distingue da altri sullo stesso gioco per la propria componente sociale, che ha un ruolo centrale nell'esperienza degli utenti. Tale componente è il risultato di varie funzionalità della web application, tra cui si evidenziano:

- **Chat con l'avversario:** nel corso della partita, i due partecipanti possono inviarsi messaggi predefiniti, anche ironici;
- **Organizzazione di tornei:** gli utenti possono organizzare tornei aperti o chiusi con quattro partecipanti;
- **Leaderboard con rating personali:** gli utenti possono confrontare i propri rating con quelli degli altri in una leaderboard generale sempre aggiornata;
- **Condivisione social:** gli utenti possono condividere il risultato della propria partita su varie piattaforme social (Whatsapp, X, Facebook).

Social Backgammon presenta anche due feature di intelligenza artificiale che rendono l'esperienza utente molto varia:

- **Avversari AI:** si possono giocare partite contro agenti AI di tre livelli di difficoltà, che possono anche essere inclusi nei tornei;
- **Suggerimenti dell'AI:** nel corso degli scontri, i giocatori possono richiedere un numero limitato di suggerimenti AI per individuare la migliore mossa da svolgere.

La combinazione di tornei, agenti AI e aspetto sociale rende Social Backgammon un prodotto unico nel mercato dei servizi di backgammon online. La componente sociale, tratto distintivo del prodotto, può attrarre una nicchia di utenti non coperta da altri servizi.

1.2 Backlog

Di seguito l'intero backlog del prodotto, ordinato per epiche:

- **Gestione dell'autenticazione**
 - Da utente, vorrei poter accedere al sito se mi sono già registrato.
 - Da utente, vorrei potermi registrare al sito se non l'ho già fatto.

- Da utente, vorrei poter cambiare la mia password se me la sono scordata.

- **Partecipazione ad una partita singola (non torneo)**

- Da utente, vorrei poter invitare un'altra persona online ad una partita.
- Da utente, vorrei poter accettare l'invito ad una partita contro un'altra persona online per iniziare la partita.
- Da utente che ha scelto le impostazioni per una partita contro l'AI, vorrei poter iniziare la partita e giocare contro l'AI.
- Da utente, vorrei che quando l'utente invitato ad una partita accetti, la partita inizi.
- Da utente, vorrei poter scegliere le impostazioni “numero di scontri per decretare la vittoria” e “difficoltà”, per giocare contro l'AI.
- Da utente che vuole mandare l'invito ad una partita ad un'altra persona online, vorrei poter scegliere il numero di scontri necessari per determinare l'esito della partita.
- Da utente, vorrei invitare una persona random ad una partita online.

- **Giocare una partita**

- Da utente che sta giocando una partita, vorrei poter commentare durante la partita.
- Da utente che sta giocando una partita, vorrei vedere un messaggio di vittoria o sconfitta a fine scontro.
- Da utente che sta giocando una partita, vorrei poter spostare la mia pedina su una pila di altre mie pedine.
- Da utente che sta giocando una partita, vorrei poter lanciare i dadi in modo da poter poi spostare le mie pedine.
- Da utente che sta giocando una partita, vorrei poter mangiare una pedina avversaria con la mia pedina.
- Da utente che sta giocando una partita, vorrei vedere la board e le pedine per giocare.
- Da utente che sta giocando una partita, vorrei poter muovere le mie pedine su una punta non occupata.
- Da utente che sta giocando una partita, quando al lancio del dado ottengo un doppio, vorrei poter muovere le mie pedine di quel numero 4 volte.
- Da utente che sta giocando una partita, se i numeri del lancio dei dadi non mi permettono di muovere alcuna pedina allora il mio turno sarà terminato.
- Da utente che sta per iniziare a giocare una partita, vorrei lanciare un dado: il giocatore che ottiene il numero maggiore farà la prima mossa (se i due numeri sono uguali il lancio si ripete).
- Da utente che sta giocando una partita, quando ho portato tutte le mie pedine nella base vorrei poter tirarle fuori dalla board in base ai numeri che mi escono dal lancio.
- Da utente che sta giocando una partita, vorrei vedere un messaggio di vittoria o sconfitta a fine partita (insieme di scontri).
- Da utente che sta giocando una partita, vorrei vedere lo stato di avanzamento della partita.
- Da utente che sta giocando una partita, vorrei avere un pulsante per abbandonare la partita e tornare al menù principale.
- Da utente che sta giocando una partita, vorrei fare gammon o backgammon.
- Da utente che sta giocando una partita, vorrei poter raddoppiare la posta in gioco tramite il dado del raddoppio per vincere il doppio dei punti con questo scontro.
- Da utente, vorrei chiedere all'AI un consiglio sulla mossa da fare.
- Da utente, vorrei che la partita concludesse in mio favore se l'avversario è inattivo.

- **Gestione del torneo**

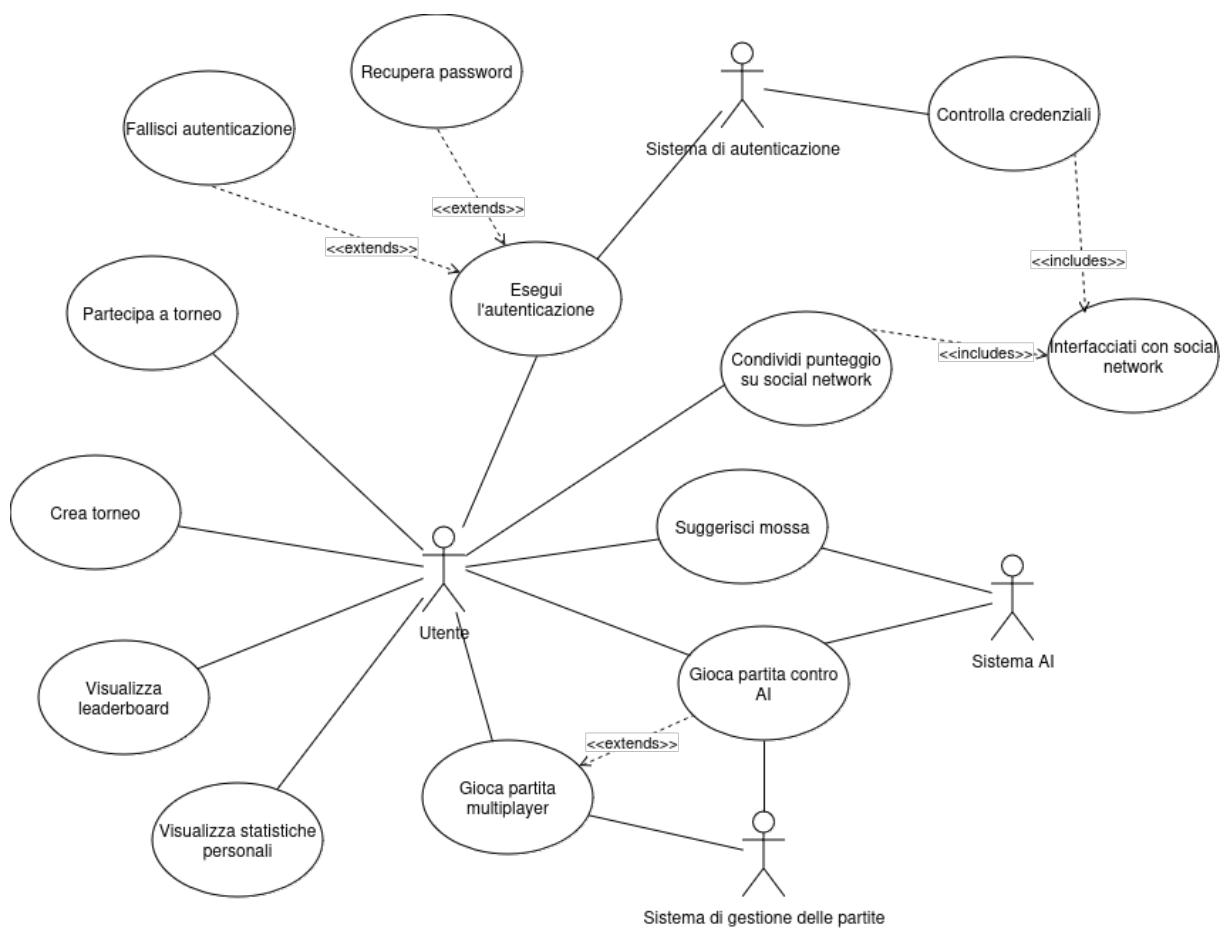
- Da utente, vorrei poter vedere i tornei a me disponibili.
- Da utente che sta configurando il suo torneo, vorrei poter limitare la partecipazione ad un torneo a utenti specifici.
- Da utente che sta configurando il suo torneo, vorrei poter rendere la partecipazione al torneo libera per qualsiasi utente.

- Da utente che sta configurando il suo torneo, vorrei poter decidere il numero di scontri per determinare la vittoria di una partita.
- Da utente, vorrei poter partecipare ai tornei a me disponibili (che vedo).
- Da utente che sta configurando il suo torneo, vorrei impostare il formato del torneo a “gironi”.

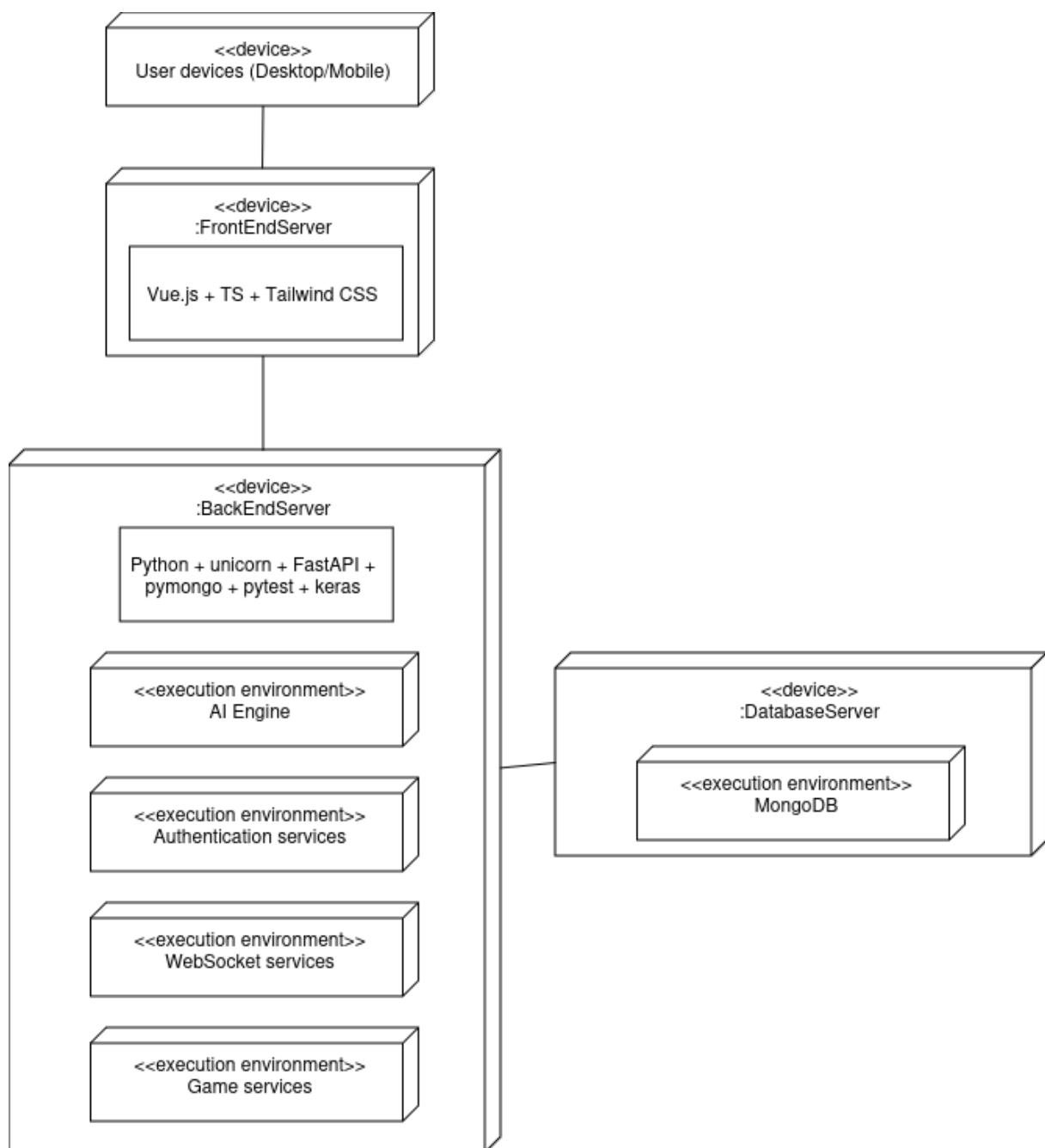
- **Gestione e visualizzazione del rating**

- Da utente, vorrei poter vedere il mio rating su una leaderboard generale.
- Da utente, vorrei poter vedere le mie statistiche.
- Da utente, vorrei che il mio rating fosse aggiornato alla fine di ogni partita in base al risultato.
- Da utente, vorrei che la leaderboard generale fosse ordinata in base al rating.

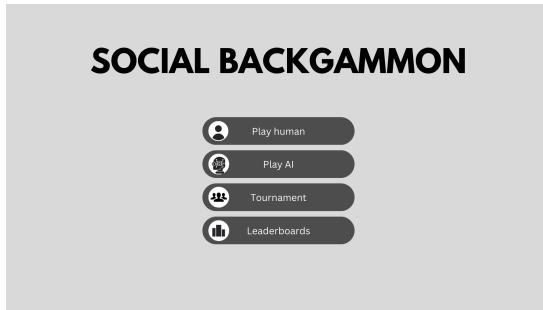
1.3 Diagramma dei casi d'uso



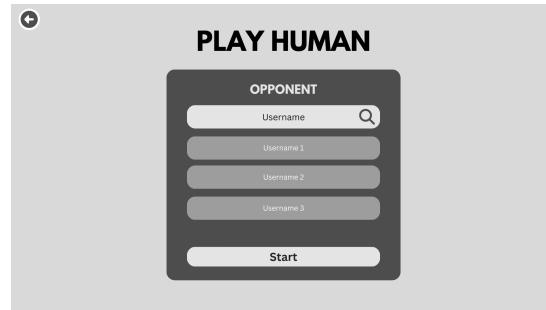
1.4 Diagramma dell'architettura



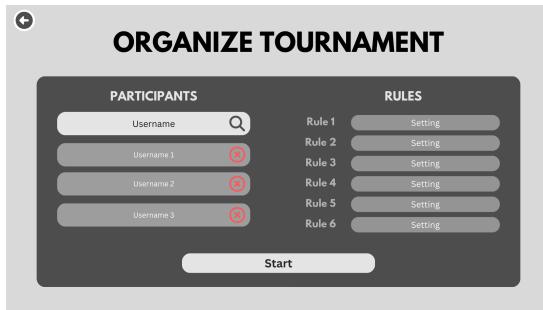
1.5 Mockup



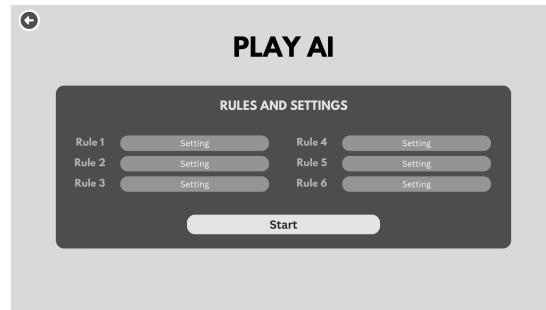
(a) Mockup della schermata home per l'accesso alle varie pagine



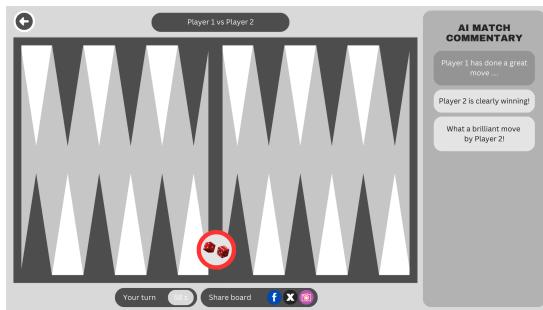
(b) Mockup della schermata di creazione di una partita contro altro utente



(c) Mockup della schermata di creazione di una partita contro AI



(d) Mockup della schermata di creazione di un torneo



(e) Mockup della schermata di gioco



(f) Mockup della leaderboard

1.6 Analisi dei rischi

I rischi possibili possono essere classificati in quattro macro-classi: rischi tecnici, rischi operativi, rischi legati all'infrastruttura e rischi di progetto.

1.6.1 Rischi tecnici

Complessità dello sviluppo AI per il gioco

- Descrizione:** Il progetto include lo sviluppo di un'intelligenza artificiale capace di giocare contro l'utente (nel caso del gioco di Backgammon o simile). L'implementazione di una AI che prenda decisioni sensate in tempo breve è complessa e potrebbe richiedere più tempo del previsto.
- Probabilità:** Alta.
- Impatto:** Alto.

- **Motivazione:** L'AI è una delle parti centrali del gioco e un avversario debole o prevedibile comprometterebbe l'intera esperienza di gioco.
- **Strategie di mitigazione:**
 - Implementare inizialmente un'AI semplificata basata su regole fisse, espandibile successivamente con modelli più avanzati.
 - Ricorrere a librerie o framework di AI già esistenti per evitare di sviluppare tutto da zero (ad esempio, algoritmi di minimax per giochi da tavolo).
 - Integrare la possibilità di tuning dell'AI per adattarne la difficoltà in modo dinamico.

Gestione del multiplayer in tempo reale

- **Descrizione:** Il multiplayer richiede una gestione efficace delle connessioni tra giocatori, della sincronizzazione degli eventi di gioco e della gestione delle disconnessioni. Implementare e ottimizzare queste funzionalità può essere complicato.
- **Probabilità:** Media.
- **Impatto:** Alto.
- **Motivazione:** Se il multiplayer non funziona correttamente, gli utenti potrebbero abbandonare il gioco a causa di disconnessioni o lag durante la partita.
- **Strategie di mitigazione:**
 - Usare protocolli WebSocket per garantire comunicazioni in tempo reale e bassa latenza.
 - Effettuare test intensivi per la gestione delle disconnessioni, assicurando che il sistema sia in grado di recuperare senza compromettere l'esperienza di gioco.
 - Salvare sul database centrale lo stato di gioco via via, in modo tale da poter recuperare il gioco in caso di disconnessione.

Integrazione con API di social network

- **Descrizione:** L'integrazione con le API di social network (per la condivisione dei risultati o il login) può risultare problematica se le API cambiano o sono soggette a limitazioni di utilizzo.
- **Probabilità:** Media.
- **Impatto:** Medio.
- **Motivazione:** La condivisione sui social è una funzionalità che aggiunge valore al progetto, ma eventuali malfunzionamenti potrebbero influenzare negativamente l'esperienza utente.
- **Strategie di mitigazione:**
 - Implementare l'integrazione in modo modulare, così da poter sostituire rapidamente una piattaforma con un'altra in caso di necessità.
 - Prevedere funzionalità alternative come salvataggio locale delle partite o dei punteggi legato all'account sulla piattaforma, in caso di indisponibilità dei social.

1.6.2 Rischi operativi

Mancanza di competenze specifiche nel team

- **Descrizione:** Il team potrebbe non avere esperienza sufficiente in alcune delle tecnologie chiave richieste dal progetto, come lo sviluppo di AI, il multiplayer o la gestione di container Docker.
- **Probabilità:** Media.
- **Impatto:** Alto.

- **Motivazione:** La mancanza di competenze potrebbe ritardare lo sviluppo o portare a soluzioni tecnicamente deboli.
- **Strategie di mitigazione:**
 - Identificare le competenze mancanti durante lo sprint 0 e organizzare sessioni di autoformazione mirate per i membri del team.
 - Considerare il riuso di codice open source o l'impiego di soluzioni preesistenti per le aree più critiche (es. modelli di AI).

Ritardi nella consegna delle funzionalità chiave

- **Descrizione:** Il mancato rispetto delle scadenze per le funzionalità più critiche, come multiplayer, AI o leaderboard, potrebbe compromettere il completamento del progetto nei tempi previsti.
- **Probabilità:** Alta.
- **Impatto:** Alto.
- **Motivazione:** Senza queste funzionalità, il prodotto non raggiungerebbe il livello minimo richiesto per essere rilasciato e testato dopo ogni sprint.
- **Strategie di mitigazione:**
 - Suddividere le funzionalità critiche in moduli incrementali che possano essere rilasciati e testati in fasi intermedie.
 - Effettuare rilasci anticipati di versioni minime funzionanti (MVP) per ottenere feedback immediato dal Requirement Owner e correggere eventuali errori prima che diventino critici.
 - Mantenere aggiornato il backlog e monitorare costantemente lo stato di avanzamento tramite strumenti come Taiga e GitLab.

1.6.3 Rischi legati all'infrastruttura

Problemi di sicurezza e vulnerabilità del codice

- **Descrizione:** Il sistema potrebbe essere soggetto a vulnerabilità di sicurezza, soprattutto se il gioco è accessibile online. Un attacco potrebbe compromettere i dati degli utenti o interrompere il servizio.
- **Probabilità:** Media.
- **Impatto:** Alto.
- **Motivazione:** La compromissione della sicurezza potrebbe esporre i dati personali dei giocatori o rendere il servizio inaffidabile, danneggiando la reputazione del prodotto.
- **Strategie di mitigazione:**
 - Usare SonarQube per analizzare continuamente il codice alla ricerca di vulnerabilità e debolezze.
 - Implementare protocolli di sicurezza robusti, come l'autenticazione OAuth per l'accesso ai social network e la crittografia delle comunicazioni (SSL/TLS).

1.6.4 Rischi di progetto

Mancanza di feedback dal Requirement Owner (Cliente)

- **Descrizione:** Un feedback limitato o ritardato da parte del Requirement Owner potrebbe portare a sviluppare funzionalità non desiderate o non allineate con le aspettative del cliente.
- **Probabilità:** Bassa.

- **Impatto:** Alto.
- **Motivazione:** La mancanza di comunicazione può causare deviazioni significative rispetto ai requisiti del prodotto, con conseguenti ritardi e revisioni del lavoro svolto.
- **Strategie di mitigazione:**
 - Pianificare revisioni periodiche (Sprint Review) con il cliente alla fine di ogni sprint per ottenere feedback tempestivo.

1.6.5 Tabella dei rischi

Rischio	Probabilità	Impatto	Mitigazione
Complessità sviluppo AI	Alta	Alto	Usare AI semplice, integrare librerie open source
Gestione multiplayer	Media	Alto	Utilizzo di WebSocket, test disconnessioni, salvataggio partita
Integrazione con API social	Media	Medio	Approccio modulare, alternative in caso di fallimento
Mancanza di competenze	Media	Alto	Autoformazione interna, riuso di codice open source
Ritardi funzionalità chiave	Alta	Alto	Sviluppo modulare, rilascio MVP, monitoraggio continuo con Taiga/GitLab
Sicurezza e vulnerabilità	Media	Alto	Analisi continua del codice con SonarQube, autenticazione sicura, crittografia SSL
Mancanza di feedback dal cliente	Bassa	Alto	Sprint review regolari

1.6.6 Conclusioni

L'analisi dei rischi ha evidenziato i principali ostacoli che potrebbero compromettere il progetto. Tuttavia, la pianificazione di strategie di mitigazione adeguate consente di affrontare tali rischi in modo proattivo. Un monitoraggio continuo e un approccio agile permetteranno di adattarsi rapidamente a eventuali problemi emergenti.

1.7 Analisi della fattibilità

La fattibilità del progetto dipende da sei macro-fattori principali: la disponibilità delle risorse, la fattibilità economica, commerciale, tecnica, legale e temporale.

1.7.1 Disponibilità delle risorse

Risorse umane Il team di progetto è composto da sei membri, una dimensione ideale per garantire un numero sufficiente di persone e un'adozione efficace del metodo Scrum. Grazie a questa metodologia, il team può gestire le attività in modo agile, con i membri che hanno assunto i ruoli previsti da Scrum.

Strumenti di sviluppo L’ambiente di sviluppo CAS garantisce accesso gratuito a tutti gli strumenti necessari per un ciclo di sviluppo efficiente, offrendo strumenti per la gestione del progetto, controllo delle versioni, analisi della qualità del codice, esecuzione automatica di test, monitoraggio del tempo di lavoro e comunicazione.

Infrastruttura tecnologica Il Dipartimento di Ingegneria e Scienze dell’Informazione (DISI) fornisce una solida infrastruttura tecnologica, che comprende hosting, server e servizi di database, offrendo tutto il necessario per supportare il progetto.

1.7.2 Fattibilità economica

Il progetto presenta un’elevata fattibilità economica per i RO poiché i costi sono praticamente nulli. Di seguito una suddivisione:

- **Costi di sviluppo:** gli strumenti impiegati sono open-source e resi disponibili gratuitamente dal DISI. Il team lavora senza compenso;
- **Costi operativi e di distribuzione:** tutti i servizi per il deployment sono forniti dal DISI e la manutenzione è gestita internamente. Gli unici costi operativi derivano dall’uso di API di social network;
- **Costi di marketing e pubblicità:** non sono previste campagne promozionali;
- **Costi di scalabilità:** eventuali spese potrebbero essere necessarie se il prodotto dovesse raggiungere un ampio pubblico globale, ma questo scenario non è previsto nel progetto iniziale.

1.7.3 Fattibilità commerciale

Esistono numerosi siti che consentono di giocare a backgammon online, sia tra due persone che contro un’intelligenza artificiale. Allo stesso modo, esistono numerose app mobili che offrono tali funzionalità, talvolta includendo anche la possibilità di partecipare a tornei. Tuttavia, non abbiamo ancora trovato una piattaforma che integri tutte queste opzioni contemporaneamente. Inoltre, molti di questi servizi non valorizzano a sufficienza l’aspetto sociale del gioco, che invece è un elemento centrale in questo prodotto.

1.7.4 Fattibilità tecnica

Pur in presenza di alcuni rischi tecnici e possibili lacune di competenze all’interno del team, la fattibilità tecnica complessiva è elevata. Le strategie di mitigazione delineate nell’analisi dei rischi affrontano i fattori di rischio, garantendo che il team possa superare con successo eventuali difficoltà tecniche.

1.7.5 Fattibilità legale

Proprietà intellettuale Se il progetto utilizza musica o immagini di terze parti, sarà necessario assicurarsi di disporre dei diritti o delle licenze per l’uso. Un’attenzione alle licenze del materiale e, dove richiesta, un’appropriata attribuzione dell’autore dovrebbero garantire il rispetto della proprietà intellettuale.

Protezione dei dati personali Considerando che il progetto prevede funzionalità online, potrebbe essere necessario raccogliere dati personali degli utenti. Il team deve impegnarsi a minimizzare la raccolta dei dati e informare gli utenti in modo trasparente. Ciò potrebbe essere complesso, vista la mancanza di conoscenze giuridiche tra gli attori coinvolti nella creazione del prodotto.

Rispetto dei termini di servizio L’utilizzo di API di social network richiede il pieno rispetto dei relativi termini di servizio. Considerando che il prodotto non richiede usi particolari delle API, il rispetto dei termini di servizio dovrebbe essere raggiungibile.

1.7.6 Fattibilità temporale

L'analisi dei rischi evidenzia possibili ritardi nello sviluppo delle funzionalità critiche, che potrebbero compromettere il completamento nei tempi previsti. Tuttavia, le strategie di mitigazione delineate, insieme all'adozione di metodologie agili, dovrebbero ridurre significativamente questi rischi. L'approccio iterativo e incrementale consente di rilasciare progressivamente le funzionalità prioritarie, migliorando la flessibilità nella gestione di eventuali ritardi o cambiamenti. Questo permette di mantenere il controllo sui tempi di consegna, minimizzando il rischio di sforamenti significativi rispetto alle scadenze stabilite.

1.7.7 Conclusioni

Il progetto ha solide basi per essere portato a termine con successo. La disponibilità delle risorse umane e tecnologiche, l'assenza di costi significativi e il supporto infrastrutturale fornito dal DISI garantiscono un'elevata fattibilità economica e operativa. Nonostante la presenza di alcuni rischi tecnici e temporali, l'adozione di metodologie agili e strategie di mitigazione consente di gestire efficacemente le criticità, minimizzando il rischio di ritardi significativi. Sul piano legale, resta l'esigenza di integrare competenze specialistiche per la gestione della protezione dei dati.

Dal punto di vista della fattibilità commerciale, il progetto si inserisce in un mercato competitivo, ma presenta elementi distintivi. La combinazione di diverse modalità di gioco e l'attenzione agli aspetti sociali offrono un valore aggiunto rispetto agli attuali prodotti esistenti. La mancanza di un servizio che integri tutte queste caratteristiche rappresenta una significativa opportunità commerciale. Se ben eseguito, il progetto ha quindi il potenziale per attrarre una nicchia di utenti interessati a un'esperienza di gioco completa e socialmente coinvolgente.

In conclusione, l'unico ostacolo alla piena realizzazione del progetto è di natura legale, poiché mancano figure con le competenze adeguate in questo ambito. Tuttavia, concentrandosi esclusivamente sugli aspetti del prodotto, si può affermare che il progetto è ben strutturato e gestibile, con ottime possibilità di essere completato nei tempi e nelle modalità previsti.

Capitolo 2

Descrizione degli sprint

2.1 Note generali

Tutti gli incrementi finali sono rilasciati come `tags` e quello finale anche come `release` per un rapido deployment. Nonostante l'incremento sia praticamente pronto per essere usato, è necessario ricordarsi di creare e inserire le variabili protette come password google e simili nei file `.env` del server e del client.

2.2 Sprint 1

2.2.1 Spint Goal

Consentire agli utenti di registrarsi e accedere al sito, oltre a implementare le funzionalità di base per iniziare e gestire una partita di Backgammon, con particolare attenzione alla visualizzazione della board, al lancio dei dadi e al movimento delle pedine. Implementare inoltre gli inviti ad una partita multigiocatore.

2.2.2 Sprint Backlog

- Da utente, vorrei potermi registrare al sito se non l'ho già fatto
- Da utente, vorrei poter accedere al sito se mi sono già registrato
- Da utente che sta giocando una partita, vorrei vedere la board e le pedine per giocare
- Da utente che sta giocando una partita, vorrei poter lanciare i dadi in modo da poter poi spostare le mie pedine
- Da utente che sta giocando una partita, vorrei poter muovere le mie pedine su una punta non occupata
- Da utente che sta giocando una partita, vorrei poter mangiare una pedina avversaria con la mia pedina
- Da utente che sta giocando una partita, vorrei poter spostare la mia pedina su una pila di altre mie pedine
- Da utente, vorrei poter invitare un'altra persona online ad una partita
- Da utente, vorrei poter accettare l'invito ad una partita contro un'altra persona online per iniziare la partita
- Da utente, vorrei che quando l'utente invitato ad una partita accetti, la partita inizi

Per più informazioni riguardo ogni user story, con i relativi punti, descrizioni, epiche e sub-tasks, controllare [Taiga](#).

2.2.3 Incremento

L'incremento realizzato nel corso dello sprint 1 è consistito nella creazione di un Minimum Viable Product che implementasse le funzionalità fondamentali dell'applicazione:

- Autenticarsi.
- Invitare un altro utente ad una partita.
- Accettare un invito ed iniziare una partita.
- Svolgimento del gioco: visualizzare il board e le pedine di gioco, lanciare i dadi, spostare le pedine secondo le regole di validità delle mosse, mangiare le pedine avversarie.

2.2.4 Definition of Done

La seguente è la lista dei requisiti definiti per lo sprint 1 per poter considerare un task come “done”:

- La funzionalità prevista dal task è stata sviluppata nel suo intero (tutti i requisiti sono stati implementati).
- Il codice non presenta errori di linting o warnings.
- Il codice è stato revisionato da almeno un altro developer e approvato.
- Il codice presenta commenti utili ad esplicarne il comportamento, dove necessario.
- I casi limite sono stati gestiti.
- Gli unit test passano e hanno una coverage di almeno l’80% della nuova funzionalità.
- È avvenuto il commit, push, e se necessario anche il merge, delle nuove modifiche su Gitlab.
- È stata scritta una documentazione minimale a riguardo.
- L’implementazione del task è stata approvata dal PO.

Nello sprint 1 la DoD è stata rispettata per tutte le US portate a completamento.

2.2.5 Esempi di test per ogni US

Di seguito, un elenco contenente un esempio di test fatto per ogni user story. I test si trovano nel [codice sorgente](#), navigando al file indicato dal percorso.

- **#2** Da utente, vorrei potermi registrare al sito se non l’ho già fatto
test: ‘should register a user’ at /client/src/tests/auth.test.ts
- **#3** Da utente, vorrei poter accedere al sito se mi sono già registrato
test: ‘should login a user’ at /client/src/tests/auth.test.ts
- **#18** Da utente che sta giocando una partita , vorrei vedere la board e le pedine per giocare
test: ‘test_custom_board_configuration’ at /server/tests/test_board_configuration.py
- **#20** Da utente che sta giocando una partita, vorrei poter lanciare i dadi in modo da poter poi spostare le mie pedine
test: ‘should fetch dice throw result and update diceResult on diceThrow method call’ at /client/src/tests/GameView.test.ts
- **#17** Da utente che sta giocando una partita, vorrei poter muovere le mie pedine su una punta non occupata
test: ‘test_move_piece’ at /server/tests/test_game.py

- **#19** Da utente che sta giocando una partita, vorrei poter mangiare una pedina avversaria con la mia pedina
test: ‘test_move_piece’ at /server/tests/test_game.py
- **#21** Da utente che sta giocando una partita, vorrei poter spostare la mia pedina su una pila di altre mie pedine
test: ‘test_move_piece’ at /server/tests/test_game.py
- **#6** Da utente, vorrei poter invitare un’altra persona online ad una partita
test: ‘test_create_invite_endpoint’ at /server/tests/test_invites.py
- **#12** Da utente, vorrei poter accettare l’invito ad una partita contro un’altra persona online per iniziare la partita
test: ‘test_receive_invite_endpoint’ at /server/tests/test_invites.py
- **#29** Da utente, vorrei che quando l’utente invitato ad una partita accetti, la partita inizi
test: ‘test_accept_invite_endpoint’ at /server/tests/test_invites.py

2.2.6 Burndown chart



Figura 2.1: Burndown chart dello sprint 1

Si noti dal burndown chart che nel giro di pochi giorni sono state completate numerose task, questo a causa di una forte similarità tra le attività ed un avanzamento parallelo di più task.

2.2.7 Retrospettiva con Essence

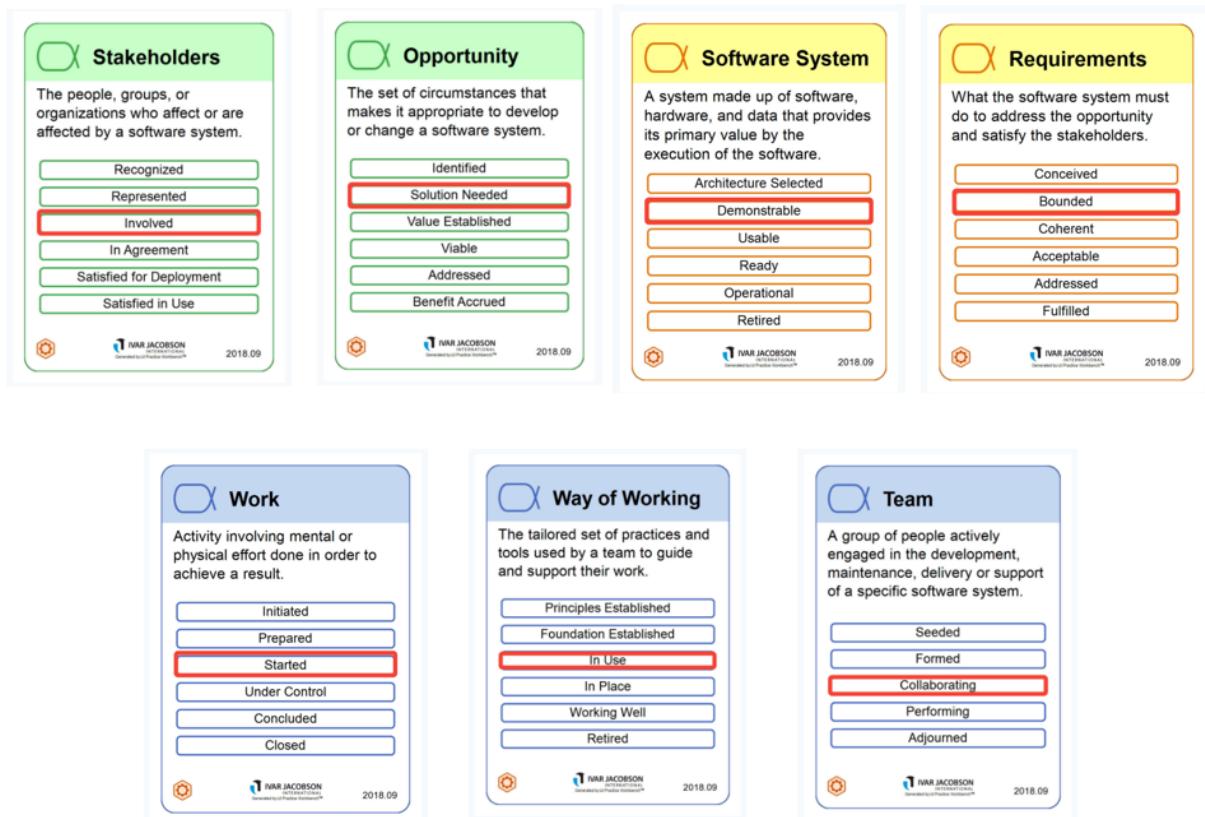


Figura 2.2: Carte essence sprint 1

• Problemi riscontrati

- Scarsa comunicazione su Mattermost
- Scarso confronto tra Developers e Product Owner
- Mancanza di organizzazione nella suddivisione delle task tra i developers

• Comportamenti adottati

- Creazione di un canale riservato al daily scrum su Mattermost
- Creazione di un canale riservato alla comunicazione tra developers e product owner su mattermost
- Decisione di assegnare in anticipo le task a ciascun developer

• Commenti

- La creazione di un canale riservato al daily scrum ha aiutato tutti ad aver ben chiari, non solo i propri ma anche quelli degli altri membri del team, i propri obiettivi della giornata. Ciò ha reso possibile un avanzamento nel progresso costante e sostenibile, inoltre ha permesso allo scrum master di monitorare il rapporto di lavoro di tutti
- La creazione di un canale riservato alla comunicazione tra developers e product owner ha permesso innanzitutto un allineamento costante tra le due parti, in più ha consentito di chiarire in modo rapido eventuali dubbi di progettazione garantendo che gli obiettivi fossero costantemente condivisi e ben compresi
- La decisione di assegnare preventivamente le task ha aiutato tanto la suddivisione del lavoro, assegnando a ciascun developer più o meno la stessa mole di lavoro, e inoltre ha garantito a ciascun membro di avere, oltre ad un obiettivo a breve termine grazie al daily scrum anche un obiettivo “finale” avendo ben chiaro fin da subito le attività da svolgere durante lo sprint attuale

2.3 Sprint 2

2.3.1 Spint Goal

Migliorare l'esperienza di gioco fornendo funzionalità chiave come la gestione dei turni, la possibilità di concludere correttamente le partite con un risultato visibile, la scelta del giocatore che inizia la partita, e l'aggiunta di interazioni di gioco come l'estrazione delle pedine, il raddoppio dei dadi e lo scambio di messaggi. Implementare il supporto al gioco contro l'AI con livelli di difficoltà personalizzabili. Aggiungere inoltre un rating ai giocatori, che si incrementa quando vincolo le partite.

2.3.2 Sprint Backlog

- Da utente che sta giocando una partita, vorrei vedere un messaggio di vittoria o sconfitta a fine scontro
- Da utente che sta giocando una partita, quando ho portato tutte le mie pedine nella base vorrei poter tirarle fuori dalla board in base ai numeri che mi escono dal lancio
- Da utente che sta giocando una partita, quando al lancio del dado ottengo un doppio, vorrei poter muovere le mie pedine di quel numero 4 volte
- Da utente che sta per iniziare a giocare una partita, vorrei lanciare un dado: il giocatore che ottiene il numero maggiore farà la prima mossa (se i due numeri sono uguali il lancio si ripete)
- Da utente che sta giocando una partita, se i numeri del lancio dei dadi non mi permettono di muovere alcuna pedina allora il mio turno sarà terminato
- Da utente che sta giocando una partita, vorrei poter commentare durante la partita
- Da utente che sta giocando una partita, vorrei vedere lo stato di avanzamento della partita
- Da utente che ha scelto le impostazioni per una partita contro l'AI, vorrei poter iniziare la partita e giocare contro l'AI
- Da utente, vorrei poter scegliere le impostazioni ‘numero di scontri per decretare la vittoria’ e ‘difficoltà’, per giocare contro l'AI
- Da utente che vuole mandare l'invito ad una partita ad un'altra persona online, vorrei poter scegliere il numero di scontri necessari per determinare l'esito della partita
- Da utente, vorrei che il mio rating fosse aggiornato alla fine di ogni partita in base al risultato

Per più informazioni riguardo ogni user story, con i relativi punti, descrizioni, epiche e sub-tasks, controllare [Taiga](#)

2.3.3 Incremento

L'incremento realizzato nel corso dello sprint 2 ha aggiunto principalmente l'implementazione di ulteriori meccaniche del gioco e la possibilità di giocare contro l'AI; in particolare:

- Meccaniche di gioco: implementazione dei meccanismi di bear-off, quadrupla mossa nel caso di valori uguali dal lancio dei dadi, assegnazione della prima mossa in base ad un lancio di dado iniziale, riconoscimento di scenari in cui nessuna mossa è possibile.
- Partite con AI: scelta delle impostazioni per una partita contro l'AI (difficoltà, numero di punti per la vittoria) e conseguente svolgimento della partita.
- Divisione della partita in più scontri, con tracciamento dello stato di avanzamento e scelta del numero di punti per decretare la vittoria della partita.
- Visualizzazione di un messaggio alla fine di ogni scontro.

- Possibilità di scambiare messaggi predefiniti con l'avversario.
- Assegnazione di un rating ad ogni giocatore e suo aggiornamento al termine di ogni partita secondo il modello ELO.

2.3.4 Definition of Done

Durante la pianificazione per lo sprint 2, vista la maggiore massa dello sprint backlog e la presenza di funzionalità potenzialmente complesse da testare (come le partite con l'AI), abbiamo deciso di diminuire il coverage minimo al 75%, consapevoli che questo avrebbe potuto portare ad una diminuzione della qualità con creazione di debito tecnico. Gli altri criteri sono rimasti invariati. Fortunatamente, questa misura cautelare si è rivelata non necessaria, e la DoD è stata rispettata per tutte le US previste dallo sprint backlog.

2.3.5 Esempi di test per ogni US

Di seguito, un elenco contenente un esempio di test fatto per ogni user story. I test si trovano nel [codice sorgente](#), navigando al file indicato dal percorso.

- **#22** Da utente che sta giocando una partita, vorrei vedere un messaggio di vittoria o sconfitta a fine scontro
test: 'shows the winner when the game is over' at /client/src/tests/GameView.test.ts
- **#83** Da utente che sta giocando una partita, quando ho portato tutte le mie pedine nella base vorrei poter tirarle fuori dalla board in base ai numeri che mi escono dal lancio
test: 'should allow moving pieces to the bear-off area if all pieces are in base' at /client/src/tests/gameService.test.ts
- **#67** Da utente che sta giocando una partita, quando al lancio del dado ottengo un doppio, vorrei poter muovere le mie pedine di quel numero 4 volte
test: 'test_throw_dice' at /server/tests/test_game.py
- **#82** Da utente che sta per iniziare a giocare una partita, vorrei lanciare un dado: il giocatore che ottiene il numero maggiore farà la prima mossa (se i due numeri sono uguali il lancio si ripete)
test: 'test_throw_start_dice' at /server/tests/test_game.py
- **#71** Da utente che sta giocando una partita, se i numeri del lancio dei dadi non mi permettono di muovere alcuna pedina allora il mio turno sarà terminato
test: 'should return null if no valid moves are available' at /client/src/tests/gameService.test.ts
- **#23** Da utente che sta giocando una partita, vorrei poter commentare durante la partita
test: 'should send preformed message on button click' at /client/src/tests/GameView.test.ts
- **#107** Da utente che sta giocando una partita, vorrei vedere lo stato di avanzamento della partita
test: 'test_round_progression' at /server/tests/test_game.py
- **#7** Da utente che ha scelto le impostazioni per una partita contro l'AI, vorrei poter iniziare la partita e giocare contro l'AI
test: 'test_move_ai' at /server/tests/test_game.py
- **#79** Da utente, vorrei poter scegliere le impostazioni 'numero di scontri per decretare la vittoria' e 'difficoltà', per giocare contro l'AI
test: 'changes difficulty and rounds_to_win values' at /client/src/tests/PlayAiView.test.ts
- **#80** Da utente che vuole mandare l'invito ad una partita ad un'altra persona online, vorrei poter scegliere il numero di scontri necessari per determinare l'esito della partita
test: 'displays the input to select the number of rounds to win' at client/src/tests/PlayHumanView.test.ts
- **#13** Da utente, vorrei che il mio rating fosse aggiornato alla fine di ogni partita in base al risultato
test: 'test_new_ratings_after_match_normal_case' at /server/tests/test_rating.py

2.3.6 Burndown chart

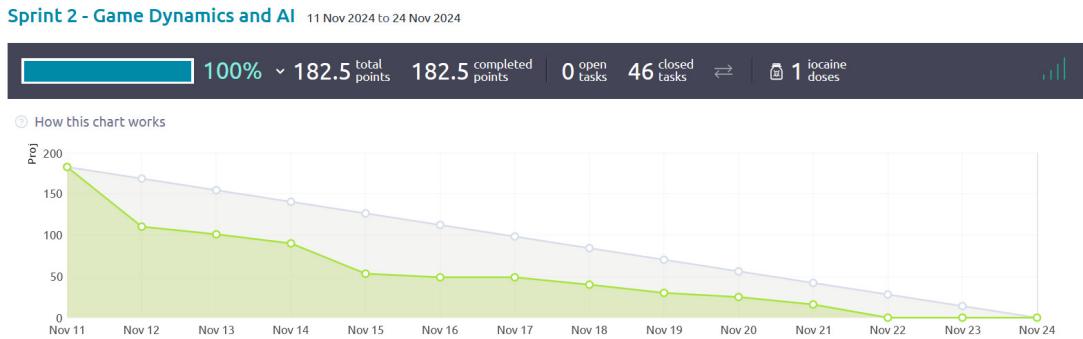


Figura 2.3: Burndown chart dello sprint 2

Il burndown chart evidenzia un andamento lineare e progressivo, segno di un lavoro costante e stabile. Si noti che il team ha lavorato in modo costante sin da inizio sprint e fino a lavoro ultimato.

2.3.7 Retrospettiva con Essence

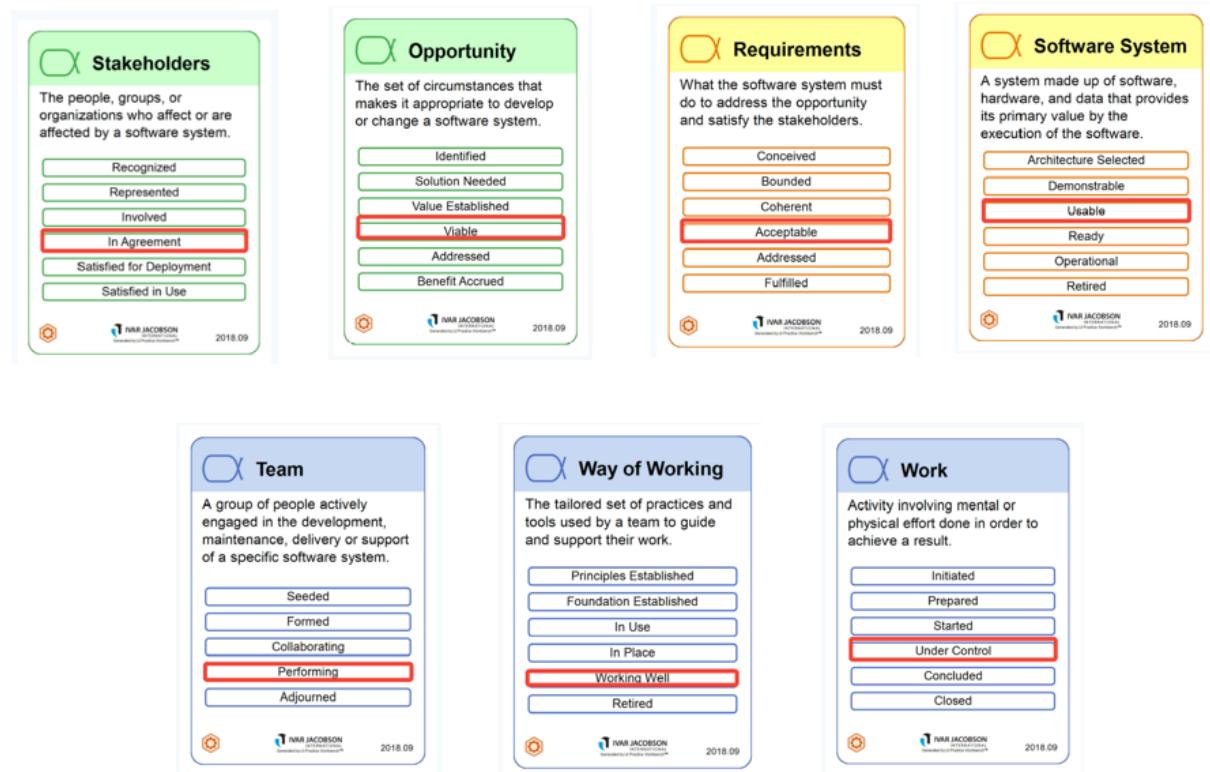


Figura 2.4: Carte essence sprint 2

- **Problemi riscontrati**

- Scarso confronto tra product owner e stakeholder
- Assenza di prove che tenessero aggiornato il cliente sull'avanzamento del prodotto

- **Comportamenti adottati**

- Organizzati maggior incontri con gli stakeholder
- Creazione di clip video delle funzionalità introdotte

- **Commenti**

- L'impegno ad organizzare maggior incontri tra product owner e stakeholder ha permesso di innanzitutto non causare alcun tipo di incertezza e disallineamento nelle aspettative del prodotto garantendo che la direzione del progetto risponda alle esigenze del cliente e soprattutto di non avere perdite di tempo, o causate appunto da incertezze che portano ad intraprendere un certo tipo di attività rivelatasi poi inadatta o non richiesta, oppure causate da dubbi concreti sulla funzionalità da implementare
- Ha migliorato decisamente la comprensione delle singole funzionalità da parte di un eventuale cliente mostrandogli effettivamente il prodotto e vedendo le proprie richieste soddisfatte e già funzionanti, garantendo un maggior coinvolgimento del cliente nel progresso del prodotto

2.4 Spint 3

2.4.1 Spint Goal

Fornire una completa esperienza di gioco multiplayer e competitiva, introducendo nuove funzionalità come i tornei, oltre a supportare l'interazione con l'AI per il suggerimento delle mosse il raddoppio della posta in gioco (in termini di rating). Inoltre, implementare la visualizzazione della classifica dei giocatori e delle statistiche personali. Per quanto riguarda la gestione del proprio account, introdurre la possibilità di recuperare la password.

2.4.2 Sprint Backlog

- Da utente, vorrei invitare una persona random ad una partita online
- Da utente che sta giocando una partita, vorrei avere un pulsante per abbandonare la partita e tornare al menù principale
- Da utente che sta giocando una partita, vorrei poter raddoppiare la posta in gioco tramite il dado del raddoppio per vincere il doppio dei punti con questo scontro
- Da utente che sta giocando una partita, vorrei fare gammon o backgammon
- Da utente che sta giocando una partita, vorrei vedere un messaggio di vittoria o sconfitta a fine partita (insieme di scontri)
- Da utente che sta configurando il suo torneo, vorrei poter decidere il numero di scontri per determinare la vittoria di una partita
- Da utente che sta configurando il suo torneo, vorrei impostare il formato del torneo a ‘gironi’
- Da utente che sta configurando il suo torneo, vorrei poter limitare la partecipazione ad un torneo a utenti specifici
- Da utente, vorrei poter cambiare la mia password se me la sono scordata
- Da utente, vorrei poter vedere i tornei a me disponibili
- Da utente, vorrei poter vedere il mio rating su una leaderboard generale
- Da utente, vorrei poter condividere il risultato della mia partita sul mio account social (e.g. Facebook)
- Da utente, vorrei poter vedere le mie statistiche
- Da utente che sta configurando il suo torneo, vorrei poter rendere la partecipazione al torneo libera per qualsiasi utente
- Da utente, vorrei poter partecipare ai tornei a me disponibili (che vedo)
- Da utente, vorrei che la leaderboard generale fosse ordinata in base al rating

- Da utente, vorrei chiedere all'AI un consiglio sulla mossa da fare
- Da utente, vorrei che la partita concludesse in mio favore se l'avversario è inattivo

Per più informazioni riguardo ogni user story, con i relativi punti, descrizioni, epiche e sub-tasks, controllare [Taiga](#)

2.4.3 Incremento

L'incremento realizzato nello sprint 3 ha riguardato principalmente l'implementazione dei tornei e del sistema di rating e leaderboard, con il completamento di alcune funzionalità precedentemente abbozzate e l'aggiunta di qualche feature che aumentasse il valore del prodotto e la sua usabilità. In particolare:

- Tornei: configurazione delle impostazioni, creazione di tornei aperti (a chiunque) e chiusi (partecipazione permessa solo per certi utenti), selezione degli utenti per tornei chiusi, visualizzazione dei tornei disponibili all'utente, partecipazione a tornei aperti e chiusi disponibili, svolgimento dei tornei secondo il formato round-robin.
- Rating e leaderboard: messaggi di visualizzazione delle modifiche ai rating a fine partita, visualizzazione della leaderboard ordinata in base al rating degli utenti tale da mostrare la top 5 e (se non incluso nella top 5) l'utente stesso, visualizzazione delle statistiche dell'utente.
- Meccaniche di gioco: implementazione del dado di raddoppio, riconoscimento di situazioni di gammon e backgammon con conseguente moltiplicazione del punteggio, abbandono della partita, meccanismo di vittoria a tavolino per timeout, richiesta di mossa consigliata all'AI.
- Implementazione di un meccanismo per permettere all'utente di sfidare un altro utente casuale.
- Reset della password dell'utente.
- Condivisione sui social del risultato della partita.
- Registrazione tramite Google.

2.4.4 Definition of Done

Durante la pianificazione per lo sprint 3, abbiamo deciso di aumentare nuovamente la percentuale minima del coverage all'80%; gli altri criteri sono invece rimasti invariati. Anche in questo sprint siamo riusciti a rispettare la DoD per ogni US prevista.

2.4.5 Esempi di test per ogni US

Di seguito, un elenco contenente un esempio di test fatto per ogni user story. I test si trovano nel [codice sorgente](#), navigando al file indicato dal percorso.

- **#132** Da utente, vorrei invitare una persona random ad una partita online
test: 'returns a random opponent username' at /client/src/tests/inviteService.test.ts
- **#169** Da utente che sta giocando una partita, vorrei avere un pulsante per abbandonare la partita e tornare al menù principale
test: 'calls the correct endpoint when confirmQuit is called' at /client/src/tests/GameView.test.ts
- **#183** Da utente che sta giocando una partita, vorrei poter raddoppiare la posta in gioco tramite il dado del raddoppio per vincere il doppio dei punti con questo scontro
test: 'calls the correct endpoint when proposeDoubling is called' at /client/src/tests/GameView.test.ts
- **#182** Da utente che sta giocando una partita, vorrei fare gammon o backgammon
test: 'should correctly determine if a gammon condition is met' at /client/src/tests/gameService.test.ts

- **#103** Da utente che sta giocando una partita, vorrei vedere un messaggio di vittoria o sconfitta a fine partita (insieme di scontri)
test: ‘renders correctly’ at /client/src/tests/MatchOverView.test.ts
- **#76** Da utente che sta configurando il suo torneo, vorrei poter decidere il numero di scontri per determinare la vittoria di una partita
test: ‘should show create panel on showCreatePanel method call’ at /client/src/tests/TournamentView.test.ts
- **#170** Da utente che sta configurando il suo torneo, vorrei impostare il formato del torneo a ‘gironi’
test: ‘test_update_tournament_stats’ at /server/tests/test_tournaments.py
- **#74** Da utente che sta configurando il suo torneo, vorrei poter limitare la partecipazione ad un torneo a utenti specifici
test: ‘test_add_participant_to_closed_tournament’ at /server/tests/test_tournaments.py
- **#5** Da utente, vorrei poter cambiare la mia password se me la sono scordata
test: ‘should set message on successful password reset’ at /client/src/tests/PasswordResetView.test.ts
- **#10** Da utente, vorrei poter vedere i tornei a me disponibili
test: ‘test_available_tournaments’ at /server/tests/test_tournaments.py
- **#11** Da utente, vorrei poter vedere il mio rating su una leaderboard generale
test: ‘renders correctly with initial data’ at /client/src/tests/LeaderboardView.test.ts
- **#14** Da utente, vorrei poter condividere il risultato della mia partita sul mio account social (e.g. Facebook)
test: ‘renders correctly’ at /client/src/tests/MatchOverView.test.ts
- **#28** Da utente, vorrei poter vedere le mie statistiche
test: ‘fetches and displays user data correctly’ at /client/src/tests/UserStatsView.test.ts
- **#75** Da utente che sta configurando il suo torneo, vorrei poter rendere la partecipazione al torneo libera per qualsiasi utente
test: ‘test_add_participant_to_open_tournament’ at /server/tests/test_tournaments.py
- **#77** Da utente, vorrei poter partecipare ai tornei a me disponibili (che vedo)
test: ‘should show join panel on showJoinPanel method call’ at /client/src/tests/TournamentView.test.ts
- **#78** Da utente, vorrei che la leaderboard generale fosse ordinata in base al rating
test: ‘renders correctly with initial data’ at /client/src/tests/LeaderboardView.test.ts
- **#199** Da utente, vorrei chiedere all’AI un consiglio sulla mossa da fare
test: ‘should call the AI suggestions endpoint’ at /client/src/tests/gameService.test.ts
- **#201** Da utente, vorrei che la partita concludesse in mio favore se l’avversario è inattivo
test: ‘should call the requestTimeout method’ at /client/src/tests/GameView.test.ts

2.4.6 Burndown chart

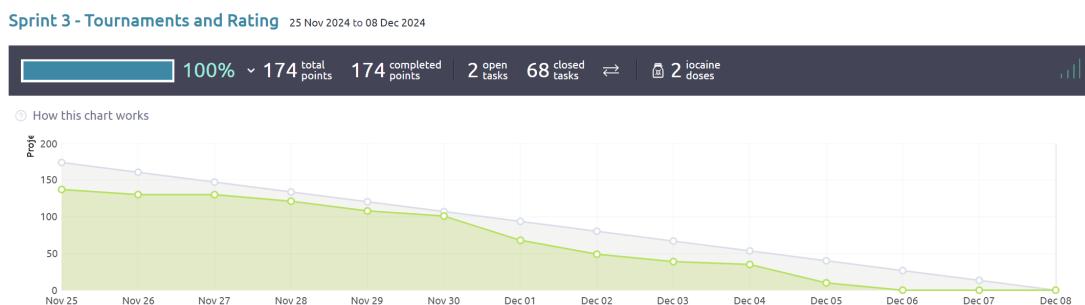


Figura 2.5: Burndown chart dello sprint 3

Il burndown chart evidenzia la costanza e stabilità del lavoro del team fin da inizio sprint, segno che le attività sono state distribuite equamente non solo tra i membri del team ma anche sul piano organizzativo individuale, ciò ha portato al raggiungimento dell'obiettivo entro le scadenze prefissate.

2.4.7 Retrospettiva con Essence

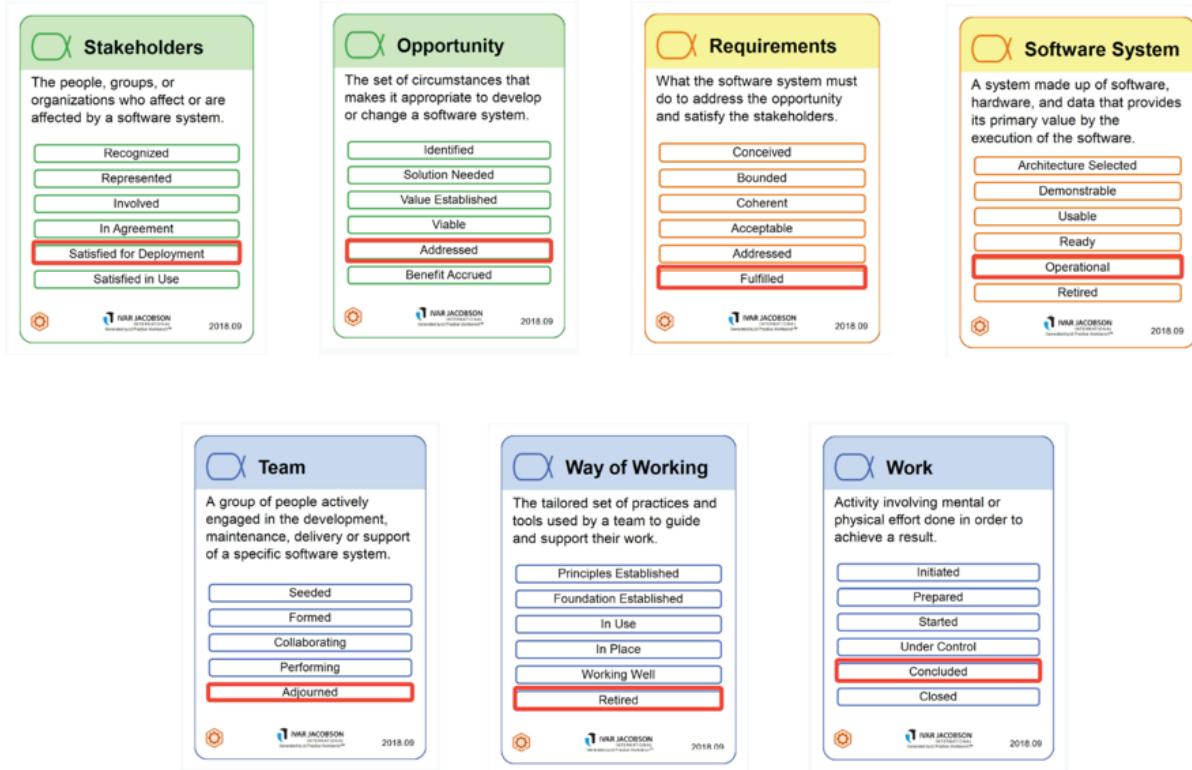


Figura 2.6: Carte essence sprint 3

• Problemi riscontrati

- Perdita di dettagli durante lo Sprint Planning
- Sottostimato il tempo di lavoro di compimento delle attività
- Tendenza ad aggiungere funzionalità non richieste

• Comportamenti adottati

- Utilizzo di template e checklist per verificare che ogni US sia sufficientemente dettagliata
- Aggiunto un tempo “cuscinetto” in caso di emergenze o imprevisti
- Verifica costante da parte dello scrum master che le attività siano in linea con i requisiti del product owner

2.5 Andamento tabelle autovalutazione



Figura 2.7: Tabella autovalutazione sprint 1



Figura 2.8: Tabella autovalutazione sprint 2

CARDS	Christian Orsi SM	Erlis Bragilevic PO	Matteo Formani DEV	Matta Ferrarini DEV	Enrico Mazzotti DEV	Lorenzo Giarrusso DEV	Motivazioni
Scrum Master	😊	😊	😊	😊	😊	😊	
Product Owner	😊	😊	😊	😊	😊	😊	
Developers	😊	😊	😊	😊	😊	😊	
Scrum Team	😊	😊	😊	😊	😊	😊	
Product Backlog	😊	😊	😐	😊	😊	😐	
Sprint Planning	😐	😐	😐	😐	😐	😐	Mancanza di cura nei dettagli (struttura e suddivisione delle US)
Sprint Goal	😊	😊	😊	😊	😊	😊	
Self Management	😊	😊	😊	😊	😐	😐	

Pienamente soddisfatto
Si può fare meglio
Per niente soddisfatto

Figura 2.9: Tabella autovalutazione sprint 3

Si noti che la tabella di autovalutazione compilata al termine del primo sprint presenta molte caselle insoddisfatte con molto margine di miglioramento in più campi a partire dallo sprint planning fino ad arrivare alla organizzazione personale del lavoro. Confrontandola con la tabella compilata al termine del terzo sprint possiamo notare un notevole miglioramento, eliminando quasi del tutto le caselle insoddisfatte questo grazie ad una continua ricerca degli errori o imprecisioni che hanno causato difficoltà durante lo sprint, andandoli a correggerli nello sprint successivo.

Capitolo 3

Descrizione del processo

3.1 Autodescrizione del team

I membri del nostro gruppo non si conoscevano tutti a vicenda, anche se alcuni di noi avevano collaborato separatamente in sottogruppi più piccoli in passato. Questo progetto ci ha aiutato a conoscerci tutti meglio, compreso coloro che avevano già collaborato in precedenza, e lavorando a stretto contatto siamo arrivati a conoscere le rispettive modalità di lavoro. Questo ci ha permesso di costruire un gruppo coeso in cui ognuno ha i propri punti di forza e abilità differenti il che aiuta a raggiungere gli obiettivi comuni con più facilità.

Il team è composto da:

- Cristian Orsi: Scrum Master
- Enis Brajević: Product Owner
- Matteo Fornaini: Developer - principalmente Dev Ops e Backend
- Mattia Ferrarini: Developer – principalmente Frontend e Backend
- Lorenzo Giarrusso: Developer – principalmente Frontend e Backend
- Enrico Mazzotti: Developer – principalmente Frontend e Backend

3.2 Risultato del team-building

Durante lo Sprint zero sono state organizzate due attività di team building attraverso i due giochi: “Scrumble” e “Escape the Boom”. Queste attività hanno permesso ai membri del team di conoscersi meglio, migliorando collaborazione, intesa e comunicazione. Le attività hanno anche portato a creare un ambiente di sviluppo positivo, fondamentale per gli sprint successivi. Concluse le attività, è stata compilata la tabella di autovalutazione del team.

GOAL	QUESTIONS	EVALUATION	Cristian Orsi (SM)	Enis Brajevic (PO)	Lorenzo Giarrusso	Matteo Fornaini	Mattia Ferrarini	Enrico Mazzotti
Learn	Q1	1 = no idea of the Scrum roles 5 = perfect knowledge of the roles and their jobs	5	5	5	5	5	5
	Q2	1 = couldn't repeat the game 5 = could play the game as a Scrum Master by himself	5	5	5	5	5	5
	Q3	1 = totally lost 5 = leads the game driving the other players	4	4	5	5	5	5
Practice	Q4	1 = feels the game is unrepeatable 5 = feels the game could be played in any situation	5	5	5	5	5	5
	Q5	1 = 0 to 3 stories 2 = 4 to 6 3 = 7 to 9 4 = 10 to 12 5 = 13 to 15	5	5	5	5	5	5
	Q6 ONLY DEV TEAM	1 = abnormal difference from the other players 5 = coherent and uniform with the group most of the time			5	5	5	5
Cooperation	Q7	1 = never speaks with the other players 5 = talks friendly to anyone in every situation	5	5	5	5	5	5
	Q8	1 = never puts effort in doing something 5 = every time is willing to understand what is going on	5	5	5	5	5	5
	Q9	1 = never asks for an opinion 5 = wants to discuss about every topic	4	5	4	5	4	5
Motivation	Q10	1 = not involved by the game 5 = always makes sure everyone is on point	5	5	5	5	5	5
	Q11 ONLY FOR PO	1 = poor/absent advices 5 = wise and helpful suggestions when is required 1 = doesn't express opinions during retrospective 5 = feels the retrospective fundamental to express		5				
	Q12		5	5	5	5	5	5
Problem Solving	Q13	<i>On the game board, if the debt pawn is on the lowest stage, the evaluation is 5, for every higher stage it decreases by 1.</i>				5		
	Q14 ONLY DEV TEAM	<i>Calculate the average of tasks left for each sprint: 1 = 21+ 2 = 16-20 3 = 11-15 4 = 6-10 5 = 0-5</i>			5	5	5	5
	Q15 ONLY FOR PO	<i>Same evaluation as Q14 for the PO</i>		5				

Figura 3.1: Tabella autovalutazione attività team-building

3.3 Sintesi dei dati

Questa sezione contiene i principali grafici relativi e tabelle relativi ai contributi del team nel corso del tempo. Sono presentati in particolare i dati ottenuti con gitinspector e quelli forniti da GitLab.

Nota: Nello sprint 1 abbiamo utilizzato lo squash dei commit al momento del merge sul main; pertanto il numero dei commit risulta essere minore nelle settimane corrispondenti. Inoltre, lo squash ha fatto sì che il commit venisse attribuito al developer che ha realizzato il merge, determinando una parziale erronea attribuzione dei contributi individuali. L'effetto di tutto ciò è particolarmente evidente nella Sezione 3.3.2.

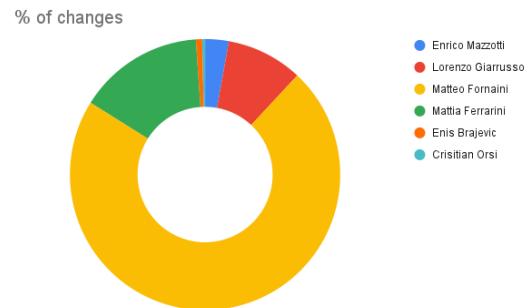
3.3.1 Dati di gitinspector

Si riportano in questa sezione le analisi di gitinspector sulle contribuzioni al repository del nostro progetto. Nella prima sezione vengono presi in esame tutti i file; mentre nella seconda si analizzano solamente i file di codice e test, ignorando quindi i file di documentazione del processo (cartella “doc/”), quelli di configurazione automaticamente generati da npm e quelli con i report di coverage. In entrambi i casi, i commit fatti dal Professor Missiroli sono ignorati. Si noti che il sistema considera l'autore del commit unico risultante (generato dal merge) la persona che ha eseguito il merge. Anche se il codice è stato scritto da altri, la cronologia mostrerà il suo nome come autore, dando l'impressione che sia stato lui a scrivere tutto il codice.

Analisi di tutti i file

Author v	Commits	Insertions	Deletions	% of changes
Cristian	6	218	1	0.35
Enis Brajevic	1	92	0	0.15
enisb1	10	354	22	0.60
Enrico Mazzotti	27	1455	328	2.82
Lorenzo Giarrusso	68	4762	1004	9.13
maforn	65	22678	9739	51.32
Matteo Fornaini	19	11356	1680	20.64
Mattia Ferrarini	58	7121	2360	15.01
Hide minor authors (3) ▾				

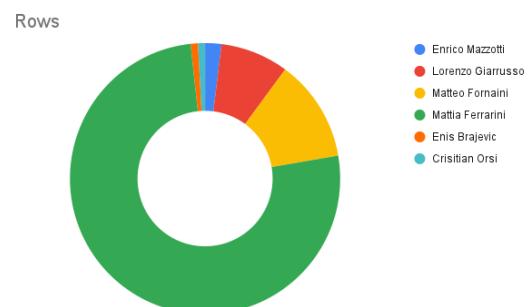
(a) Tabella dei commit



(b) Grafico dei changes

Author v	Rows	Stability	Age	% in comments
Cristian	323	148.2	4.6	0.31
Enis Brajevic	5	5.4	0.0	0.00
enisb1	328	92.7	2.4	0.00
Enrico Mazzotti	744	51.1	1.0	0.00
Lorenzo Giarrusso	3111	65.3	3.5	0.80
maforn	28995	127.9	2.6	0.21
Mattia Ferrarini	4649	65.3	2.9	0.06
Hide minor authors (3) ▾				

(c) Tabella delle righe sopravvissute



(d) Grafico delle righe sopravvissute

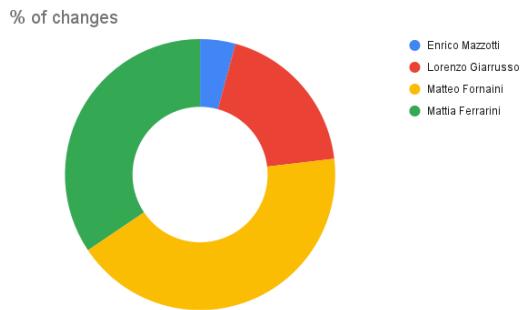


(e) Timeline

Analisi dei file di codice e test

Author	Commits	Insertions	Deletions	% of changes
Enrico Mazzotti	24	937	172	4.23
Lorenzo Giarrusso	64	4084	877	18.91
maforn	56	4237	1599	22.25
Matteo Fornaini	13	3865	1431	20.19
Mattia Ferrarini	54	6715	2316	34.43

(a) Tabella dei commit



(b) Grafico dei changes

Author	Rows	Stability	Age	% in comments
Enrico Mazzotti	501	53.5	1.3	0.00
Lorenzo Giarrusso	2593	63.5	3.0	0.96
maforn	5830	137.6	2.3	0.99
Mattia Ferrarini	4336	64.6	2.5	0.05

(c) Tabella delle righe sopravvissute



(d) Grafico delle righe sopravvissute



(e) Timeline

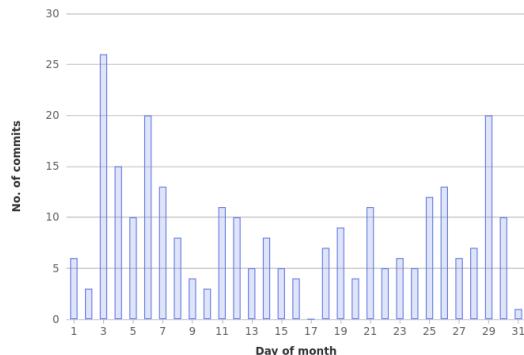
3.3.2 Distribuzione temporale dei commit

In totale, ci sono stati 267 commit al main, per una media di 4.7 al giorno. Si riportano in questa sezione alcuni grafici significativi relativi alla distribuzione temporale dei commit. Tali grafici sono presenti nelle sezioni “Analytics > Repository” e “Repository > Contributor Statistics” di GitLab.

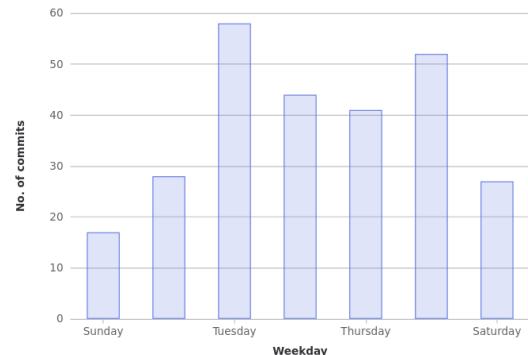
Distribuzione dei commit del team nel tempo



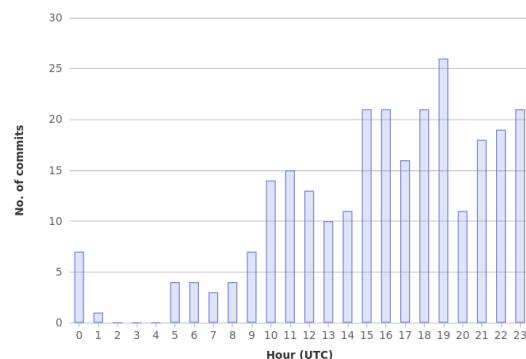
(a) Distribuzione nel periodo di lavoro



(b) Distribuzione nei giorni del mese

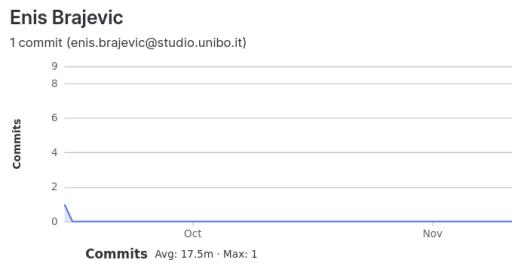
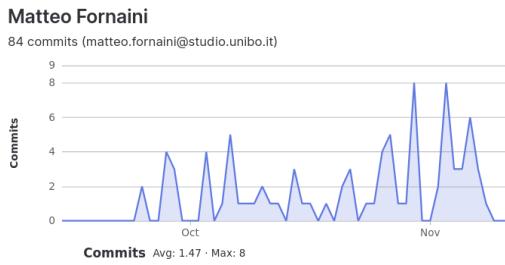


(c) Distribuzione nei giorni della settimana



(d) Distribuzione nelle ore del giorno

Distribuzione dei commit individuali nel tempo



3.4 Descrizione dell'utilizzo degli strumenti di comunicazione

Per garantire una comunicazione costante ed efficiente nel team abbiamo utilizzato diversi strumenti di comunicazione a seconda del tipo di comunicazione che ci dovevamo scambiare. Per quanto riguarda la collaborazione quotidiana abbiamo utilizzato maggiormente Mattermost, il quale grazie alla creazione di diversi canali ci ha permesso di incanalare certi tipi di conversazioni/comunicazioni nei canali riservati permettendo di intraprendere più conversazioni nello stesso tempo senza rischiare di fare confusione. Abbiamo creato, come già detto, più canali:

- Ask PO: riservato alle domande riguardanti dubbi di realizzazione del prodotto o per qualsiasi altro tipo di dubbio tra i developers e il product owner

- Backend: riservato a domande riguardanti la realizzazione del backend del prodotto, utilizzato maggiormente dai developers
- Daily Scrum: riservato alla comunicazione giornaliera degli obiettivi ed eventuali problemi riscontrati durante il giorno precedente da parte dei developers
- Database: riservato a domande relative alla struttura/realizzazione del database
- Frontend: riservato a domande riguardanti la realizzazione del frontend del prodotto, utilizzato maggiormente dai developers

Per quanto riguarda invece comunicazioni che differiscono da uno dai temi precedenti si è utilizzato WhatsApp grazie alla sua comodità e alla presenza di notifiche push (non presenti in mattermost) permette di scambiare facilmente opinioni e dubbi, ottenendo risposte pressoché immediate. Infine si sono utilizzati anche i commenti su gitlab per le merge request, con lo scopo di scambiare messaggi brevi e concisi che permettessero però la comprensione di tutti.

3.5 Descrizione dell'utilizzo di LLM

Abbiamo utilizzato gli LLM in tre modi principali:

- **Supporto alla creazione di artefatti e scrittura di documenti:** Nel corso dello sprint 0 abbiamo utilizzato ChatGPT come aiuto nella scrittura di user story e nello svolgimento delle analisi di rischi e fattibilità;
- **Supporto alla scrittura di test:** Abbiamo utilizzato ChatGPT per la generazione di funzioni di testing, che però abbiamo dovuto spesso integrare.
- **Supporto all'inizializzazione di nuove funzionalità:** Abbiamo utilizzato Copilot per creare versioni di base di nuove funzionalità.

Esempi dei prompt e delle risposte che abbiamo ottenuto dagli LLM nei contesti descritti sopra sono presenti nell'appendice A.

Abbiamo inoltre utilizzato Copilot come supporto alla scrittura del codice. In particolare, è stata molto utile la sua funzione di autocompletamento. La sua utilità è aumentata con il tempo, dato che ci ha permesso di evitare di dover fare manualmente alcune attività ripetitive, come la scrittura di base di un endpoint in FastAPI o di un servizio di update del database.

3.6 Schermata finale di SonarQube sul codice prodotto

SonarQube e SonarLint sono stati estensivamente usati durante lo sviluppo del prodotto, anche se abbiamo riscontrato diverse difficoltà con il setup di SonarQube, dato che alcune impostazioni per il parsing della coverage non erano impostabili senza permessi da amministratore, permessi non concessi agli utenti degli studenti su aminsep. Abbiamo risolto questo problema cambiando manualmente le coverage generate da pytest (questo problema, infatti, si presentava solo con i test python, non con quelli javascript). Nonostante i problemi riscontrati durante il percorso, abbiamo raggiunto degli ottimi risultati nella valutazione di SonarQube. Questo è il riassunto generale dei risultati, prima di passare ad analizzarli nel dettaglio.

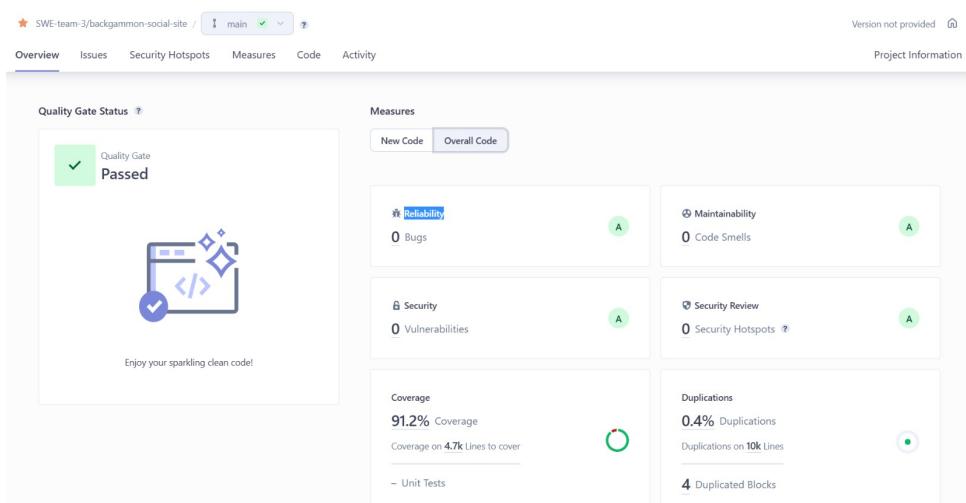
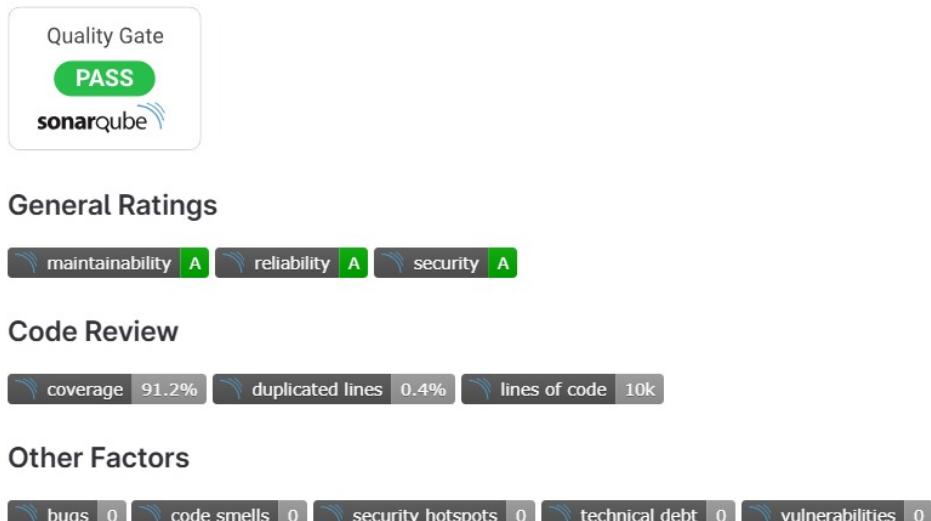


Figura 3.6: Metriche di qualità di SonarQube

Ora analizziamo nel dettaglio ognuno di questi elementi:

- **Quality Gate:** Il codice soddisfa tutti gli standard di qualità definiti, passando la quality gate.
- **Reliability, Maintainability, Security, Security Review:** abbiamo guadagnato il massimo score conseguibile (A) in ognuna di queste quattro categorie, dato che abbiamo risolto ogni possibile problema trovato da SonarQube.
- **Duplications:** su più di diecimila righe di codice le nostre duplicazioni sono segnate allo 0.4%, una cifra indubbiamente negligibile. Avremmo potuto ridurre allo 0.0% le duplicazioni, ma abbiamo deciso di non farlo in quanto non eravamo d'accordo con la segnalazione di SonarQube: questi 4 blocchi duplicati sono parte di un test che deve fare azioni simili ma non uguali per controllare la ricezione dei websocket. Estrarre una parte di codice simile per creare una funzione parametrizzata ci pareva complicasse la comprensione del codice, e quindi abbiamo deciso di lasciarlo così.
- **Coverage:** la code coverage è ben al 91.2%, che su più di diecimila righe di codice è un ottimo risultato. Per diciotto file la coverage raggiunge anche il 100%, mentre molti altri hanno valori vicini al 99%, mancando solo qualche `if` non coperto. Gli unici file che hanno una coverage dell'82% circa sono quelli lato server che devono gestire sempre l'eccezione nel caso in cui l'utente non sia autenticato o non abbia nessuna partita iniziata. In quei casi testiamo i casi limite solo un paio di volte e lasciamo le funzioni con la stessa logica ma in punti diversi del codice non testate, altrimenti i test sarebbero diventati troppo grossi.

Analizziamo infine il grafico dei problemi rispetto al tempo secondo SonarQube.

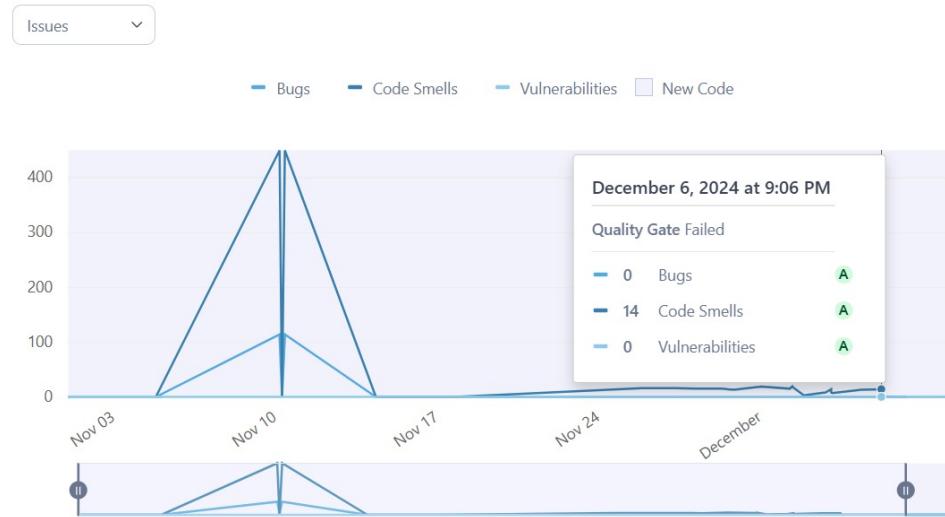


Figura 3.7: Grafico dei problemi rispetto al tempo

C'è un evidente picco iniziale con più di 400 issues, ma quello è solo stato un errore di configurazione per cui SonarQube è andato a controllare anche i file dentro "node_modules" e dentro il virtual environment di python, quindi aveva scannerizzato anche tutte le dipendenze. Rimosse quelle cartelle e fixata la pipeline possiamo vedere come ciclicamente via via col tempo le issues aumentano, fino a quando non vengono aggiustate dai programmatori per non accumulare debito tecnico. Nell'ultimo sprint abbiamo risolte le ultime 14 issues rimanenti, rendendo così il codice completamente privo di problemi e guadagnandoci il rating A per tutte le categorie.

3.7 Diagramma di deployment del prodotto

Il deployment del prodotto può avvenire in tre modalità principali diverse, in base alle esigenze dell'utente.

1. Deployment in produzione Per quanto riguarda il deployment su un server per la distribuzione con la stessa porta per client e server il diagramma di deployment è il seguente:

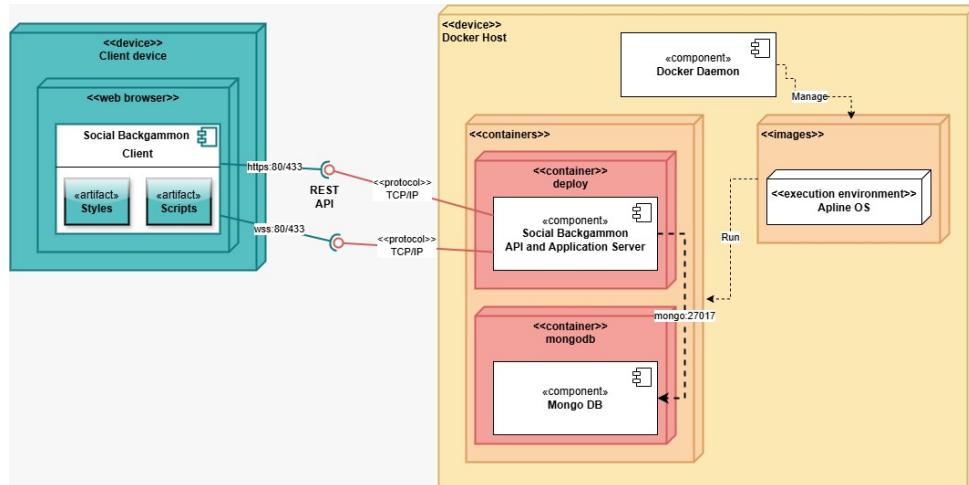


Figura 3.8: Diagramma di deployment in produzione

2. Deployment in sviluppo con docker Per quanto riguarda invece il deployment in sviluppo con docker compose il diagramma è leggermente diverso, in quanto non usa una porta singola e usa due container diversi uno per il client e l'altro per il server.

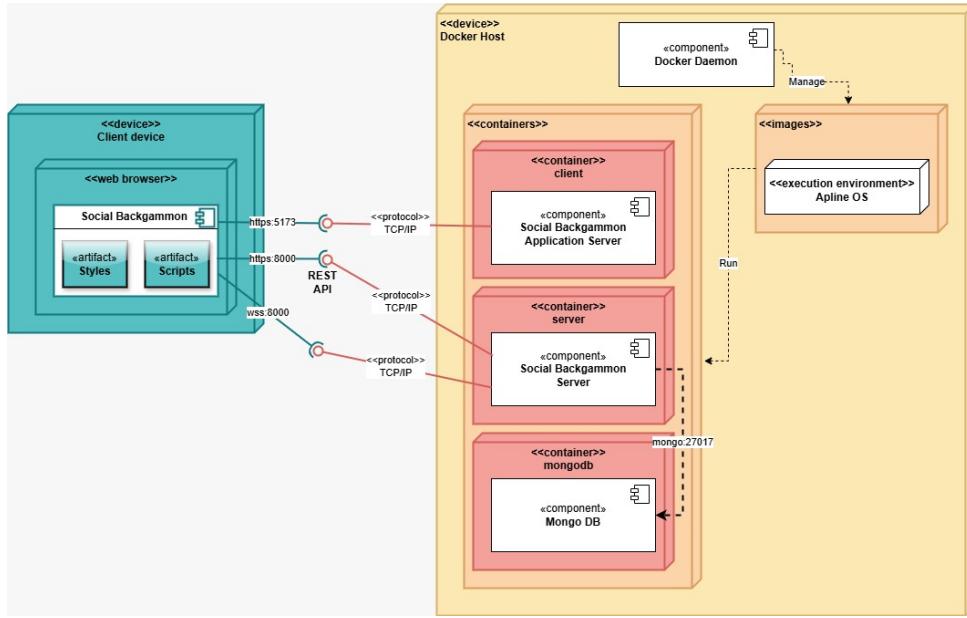


Figura 3.9: Diagramma di deployment in sviluppo con docker

3. Deployment in sviluppo senza docker Per quanto riguarda invece il deployment in sviluppo senza docker il diagramma riporta la soluzione di servizi fatti partire senza docker per il testing.

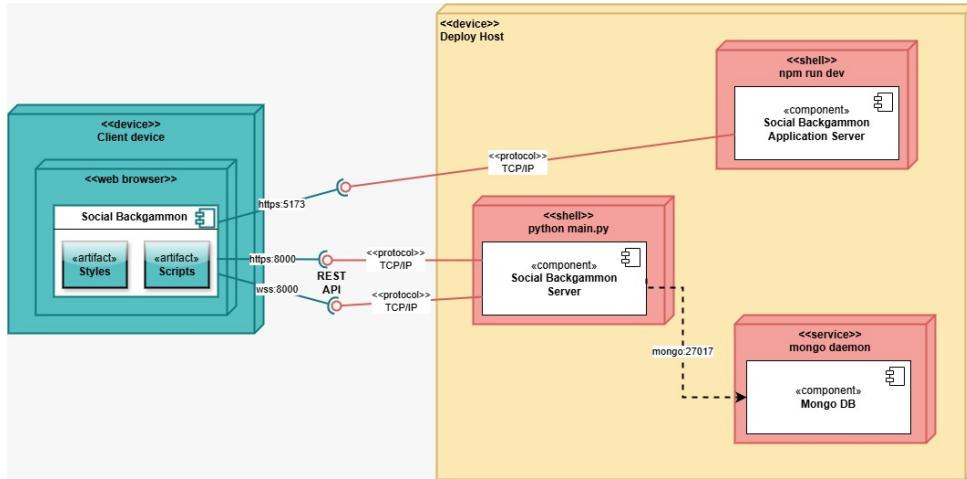


Figura 3.10: Diagramma di deployment in sviluppo senza docker

3.8 Continuous integration e continuous development

Non potendo usare Jenkins su Aminsep senza avere un'istanza in locale, il nostro team ha deciso di usare le gitlab pipeline con dei runners creati e impostati su un server privato. Durante questi tre sprint di sviluppo sono state eseguite più di 600 pipelines, di cui circa 400 passate e 200 fallite, per un totale di più di 2100 jobs. I runner con docker sopra hanno usato più di 140GB di memoria fisica e hanno eseguito codice per più di 3100 minuti (52 ore!).

La pipeline utilizzata si compone di due o quattro jobs in base al branch su cui è eseguita che si compongono sempre di due jobs per il check di SonarQube (`sonarqube-check` e `sonarqube-vulnerability-report`). Inoltre per i push sul main e sulle merge requests vengono eseguiti altri due jobs fondamentali per

controllare l'integrità del codice che sono build e test. Questi due jobs si assicurano che prima di avere cambiamenti sul main i file riescano comunque ad essere buildati correttamente per il deploy e che passino i test.

3.9 Retrospettiva finale con Essence

Le carte essence che abbiamo utilizzato per descrivere lo stato del nostro lavoro sono state:

- Le sette carte Alpha
- Definition of Done
- Product Goal
- Sprint Goal
- Sprint
- Sprint Planning
- Sprint Retrospective
- Product Backlog
- Sprint Review
- Sprint Backlog
- Daily Scrum
- Product Owner
- Developers
- Scrum Master
- Scrum Team
- Focus
- Respect
- Self-Management
- Commitment

Abbiamo scelto queste carte oltre alle “principali” perchè rispecchiavano aspetti o ruoli che dovevano essere migliorati, e questo ci ha aiutato a tenerli, innanzitutto, sempre monitorati e a notarne eventuali miglioramenti o peggioramenti durante l'avanzamento della collaborazione.

Di seguito è riportato lo stato di avanzamento del processo attraverso le checklist presenti sulle alpha card essence:

- Customer
 - Stakeholders



Satisfied for Deployment: Il prodotto risponde ai requisiti stabiliti e riteniamo che sia pronto per essere distribuito.

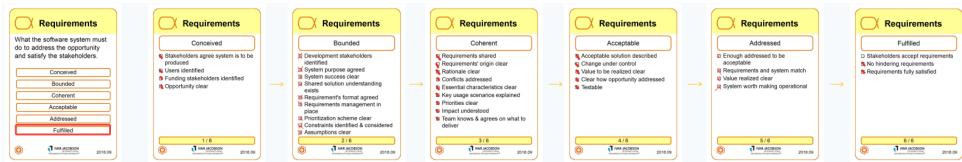
– Opportunity



Addressed: L'opportunità è stata concretamente affrontata attraverso lo sviluppo della soluzione. Il prodotto risponde all'opportunità identificata e agli obiettivi concordati.

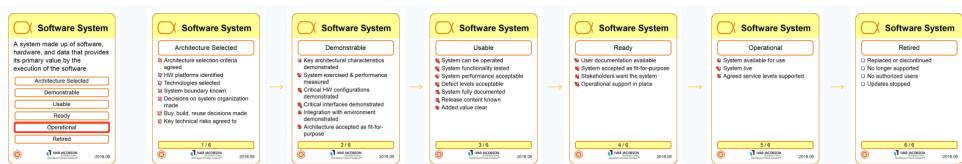
• Solution

– Requirements



Fulfilled: I requisiti sono stati pienamente implementati e verificati nel prodotto finale.

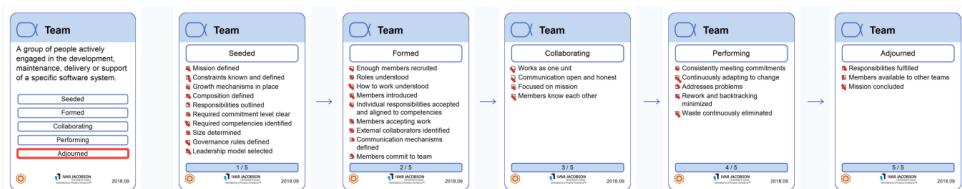
– Software System



Operational: Il sistema è attivo e utilizzato in produzione. Il software supporta gli utenti finali in scenari di utilizzo reale.

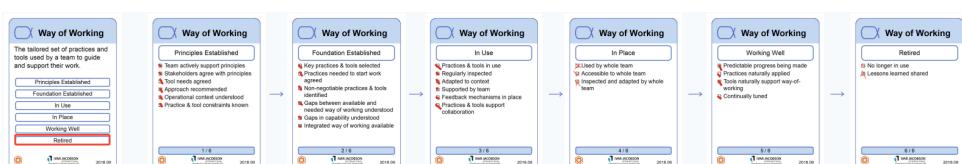
• Endeavour

– Team



Adjourned: Il team ha completato il lavoro per cui è stato formato.

– Way of Working



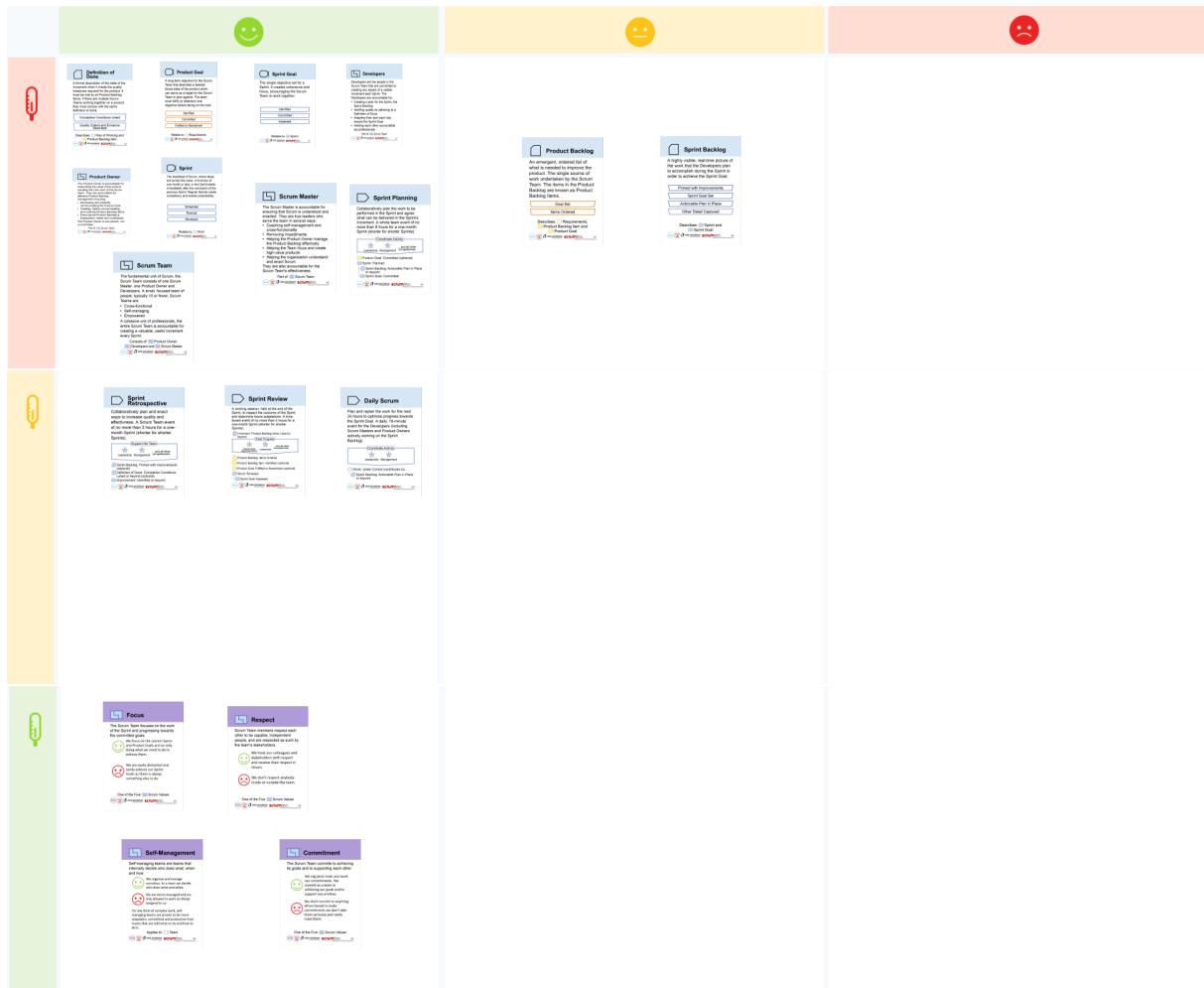
Retired: La way of working è stata dismessa perché il progetto è stato terminato.

- Work



Concluded: Lo sprint finale è stato completato. La sprint review e la retrospettiva sono state condotte.

Tabella retrospettiva finale



3.10 Pratiche agili utilizzate

Durante lo sviluppo del prodotto sono state rispettate le pratiche del metodo Agile per favorire uno sviluppo rapido, flessibile e collaborativo. Le pratiche agili adottate sono state:

- **Sviluppo iterativo e incrementale:** suddividendo il lavoro in brevi cicli chiamati sprint, inoltre ogni iterazione produce un incremento del prodotto funzionante. Per garantire un'integrazione continua sono stati fondamentali strumenti come GitLab.
- **Collaborazione e coinvolgimento del cliente:** i clienti sono coinvolti durante l'intero processo per fornire feedback continui e assicurare che il prodotto soddisfi le loro aspettative.
- **Collaborazione e comunicazione nel team:** attraverso processi come il daily scrum, lo sprint planning, la sprint review e la sprint retrospective è stato possibile sincronizzare in modo efficiente il lavoro del team pianificando e analizzando il lavoro da fare o il lavoro già fatto. Per garantire una comunicazione ottimale si sono utilizzati strumenti come Mattermost, in grado di consentire uno scambio di informazione istantaneo.
- **Gestione del progetto:** grazie ad una lista prioritaria e dinamica delle funzionalità da implementare ovvero il backlog, aggiornato costantemente in base ai feedback del cliente dal Product Owner, ed ai burndown chart che hanno permesso di monitorare il progresso del lavoro durante il singolo sprint. La gestione del progetto è stata resa possibile da strumenti come Taiga in grado di monitorare tutti i dati relativi all'avanzamento del processo.
- **Pair programming:** si è attuato il pair programming facendo lavorare due sviluppatori sul medesimo codice, questo per migliorarne la qualità e tenere tutti i componenti del team aggiornati, inoltre il codice è stato regolarmente rivisto per garantirne la qualità.
- **Controllo di versione:** l'uso di Git ha permesso lo sviluppo di più funzionalità in parallelo grazie alla gestione tramite branch, inoltre ha favorito un'integrazione continua grazie alla possibilità di fondere il nuovo codice nel branch principale. Riassumendo Git è stato essenziale per gestire il versionamento del codice sorgente, consentendo al team di avanzare senza conflitti.
- **Analisi statica del codice:** grazie all'utilizzo dello strumento SonarQube è stato possibile identificare problemi di qualità del codice, eventuali violazioni delle regole di stile oppure vulnerabilità di sicurezza.

Capitolo 4

Demo

4.1 Link alla demo

La demo è presente al seguente link: [DEMO](#). Offre un'introduzione visiva, con audio esplicativo, del prodotto realizzato. In soli sei minuti, la demo mira a far vedere allo stakeholder le caratteristiche principali del prodotto realizzato.

Capitolo 5

Liste e descrizione degli artefatti

5.1 Product Backlog

Il product backlog è un documento che contiene un elenco organizzato di tutte le funzionalità, miglioramenti e correzioni che potrebbero servire al prodotto. Contiene user stories, epiche e requisiti.

Su [taiga](#) è presente nel completo, ma lo riporto anche qua, ordinato per Epiche:

- **Gestione dell'autenticazione**
 - Da utente, vorrei poter accedere al sito se mi sono già registrato
 - Da utente, vorrei potermi registrare al sito se non l'ho già fatto
 - Da utente, vorrei poter cambiare la mia password se me la sono scordata
- **Partecipazione ad una partita singola (non torneo)**
 - Da utente, vorrei poter invitare un'altra persona online ad una partita
 - Da utente, vorrei poter accettare l'invito ad una partita contro un'altra persona online per iniziare la partita
 - Da utente che ha scelto le impostazioni per una partita contro l'AI, vorrei poter iniziare la partita e giocare contro l'AI
 - Da utente, vorrei che quando l'utente invitato ad una partita accetti, la partita inizi
 - Da utente, vorrei poter scegliere le impostazioni ‘numero di scontri per decretare la vittoria’ e ‘difficoltà’, per giocare contro l'AI
 - Da utente che vuole mandare l'invito ad una partita ad un'altra persona online, vorrei poter scegliere il numero di scontri necessari per determinare l'esito della partita
 - Da utente, vorrei invitare una persona random ad una partita online
- **Giocare una partita**
 - Da utente che sta giocando una partita, vorrei poter commentare durante la partita
 - Da utente che sta giocando una partita, vorrei vedere un messaggio di vittoria o sconfitta a fine scontro
 - Da utente che sta giocando una partita, vorrei poter spostare la mia pedina su una pila di altre mie pedine
 - Da utente che sta giocando una partita, vorrei poter lanciare i dadi in modo da poter poi spostare le mie pedine
 - Da utente che sta giocando una partita, vorrei poter mangiare una pedina avversaria con la mia pedina
 - Da utente che sta giocando una partita, vorrei vedere la board e le pedine per giocare
 - Da utente che sta giocando una partita, vorrei poter muovere le mie pedine su una punta non occupata

- Da utente che sta giocando una partita, quando al lancio del dado ottengo un doppio, vorrei poter muovere le mie pedine di quel numero 4 volte
- Da utente che sta giocando una partita, se i numeri del lancio dei dadi non mi permettono di muovere alcuna pedina allora il mio turno sarà terminato
- Da utente che sta per iniziare a giocare una partita, vorrei lanciare un dado: il giocatore che ottiene il numero maggiore farà la prima mossa (se i due numeri sono uguali il lancio si ripete)
- Da utente che sta giocando una partita, quando ho portato tutte le mie pedine nella base vorrei poter tirarle fuori dalla board in base ai numeri che mi escono dal lancio
- Da utente che sta giocando una partita, vorrei vedere un messaggio di vittoria o sconfitta a fine partita (insieme di scontri)
- Da utente che sta giocando una partita, vorrei vedere lo stato di avanzamento della partita
- Da utente che sta giocando una partita, vorrei avere un pulsante per abbandonare la partita e tornare al menù principale
- Da utente che sta giocando una partita, vorrei fare gammon o backgammon
- Da utente che sta giocando una partita, vorrei poter raddoppiare la posta in gioco tramite il dado del raddoppio per vincere il doppio dei punti con questo scontro
- Da utente, vorrei chiedere all'AI un consiglio sulla mossa da fare
- Da utente, vorrei che la partita concludesse in mio favore se l'avversario è inattivo

- **Gestione del torneo**

- Da utente, vorrei poter vedere i tornei a me disponibili
- Da utente che sta configurando il suo torneo, vorrei poter limitare la partecipazione ad un torneo a utenti specifici
- Da utente che sta configurando il suo torneo, vorrei poter rendere la partecipazione al torneo libera per qualsiasi utente
- Da utente che sta configurando il suo torneo, vorrei poter decidere il numero di scontri per determinare la vittoria di una partita
- Da utente, vorrei poter partecipare ai tornei a me disponibili (che vedo)
- Da utente che sta configurando il suo torneo, vorrei impostare il formato del torneo a ‘gironi’

- **Gestione e visualizzazione del rating**

- Da utente, vorrei poter vedere il mio rating su una leaderboard generale
- Da utente, vorrei poter vedere le mie statistiche
- Da utente, vorrei che il mio rating fosse aggiornato alla fine di ogni partita in base al risultato
- Da utente, vorrei che la leaderboard generale fosse ordinata in base al rating

5.2 Sprint Backlog

Lo sprint backlog è una selezione di elementi dal Product Backlog che il team si impegna a completare durante lo sprint, insieme a un piano su come farlo.

- Sprint 1 backlog
- [Sprint 1 su Taiga](#)
- Sprint 2 backlog
- [Sprint 2 su Taiga](#)
- Sprint 3 backlog
- [Sprint 3 su Taiga](#)

5.3 Incremento

L'incremento è la somma di tutti gli elementi completati del Product Backlog durante lo sprint corrente e quelli precedenti. Deve essere funzionante e pronto per un potenziale rilascio.

- Incremento sprint 1
- Incremento sprint 2
- Incremento sprint 3
- [Incremento su GitLab](#)

5.4 Burndown Chart

- Burndown chart sprint 1
- Burndown chart sprint 2
- Burndown chart sprint 3
- [Burndown chart su Taiga](#)

5.5 Definition of Done (DoD)

Una lista di controllo che specifica quando un'attività può essere considerata “fatta”. Include criteri di accettazione, test, documentazione e rilascio.

- DoD sprint 1
- [DoD sprint 1](#)
- DoD sprint 2
- [DoD sprint 2](#)
- DoD sprint 3
- [DoD sprint 3](#)

Capitolo 6

Prodotto live

6.1 Prodotto live

Il prodotto live può essere usato e testato su backgammon.hopto.org.

Per accedere è possibile creare nuovi utenti o fare il login con Google. Le varie funzionalità sono tutte online, anche se per giocare multigiocatore è necessario avere due utenti diversi e giocare scambiando via via gli utenti.

Appendice A

Prompt e risposte degli LLM

A.1 Supporto alla creazione del Product Backlog

PO

Quali sono le user stories per questo prodotto: Backgammon social site. Realizzazione di un prodotto (es. web application) che permetta di giocare a backgammon online tra due persone o tra persona e un “agente intelligente”. Il prodotto deve permettere di organizzare tornei fino a 4 giocatori e anche di tenere una classifica aggiornata (es. leaderboard) basata sui rating personali.

ChatGPT

Per il prodotto **Backgammon social site**, ecco alcune user stories che potrebbero descrivere le principali funzionalità:

1. Registrazione e autenticazione utente

- Come utente, voglio potermi registrare e accedere al sito con il mio account personale, in modo da partecipare a partite e tornei.
- Come utente, voglio poter collegare il mio account social (ad esempio Google o Facebook) per accedere più velocemente.

2. Creazione e gestione di partite

- Come utente, voglio poter sfidare un'altra persona online a una partita di backgammon, in modo da giocare una partita in tempo reale.
- Come utente, voglio poter giocare contro un agente intelligente (AI) per allenarmi o giocare quando non ci sono altri giocatori disponibili.

3. Organizzazione di tornei

- Come utente, voglio poter organizzare tornei con un massimo di 4 giocatori, in modo da competere con i miei amici o altri utenti.
- Come utente, voglio visualizzare il calendario e il tabellone dei tornei, così posso sapere quando è il mio turno e contro chi devo giocare.

4. Sistema di leaderboard e rating

- Come utente, voglio vedere la mia posizione nella classifica globale basata sul mio rating personale, così posso confrontarmi con gli altri giocatori.
- Come utente, voglio che il mio rating sia aggiornato automaticamente alla fine di ogni partita o torneo, in modo che il sistema rifletta accuratamente le mie prestazioni.

5. Comunicazione tra giocatori

- Come utente, voglio poter inviare messaggi al mio avversario durante la partita, così possiamo comunicare e interagire.

- Come utente, voglio avere accesso a una chat di gruppo per i tornei, così posso parlare con gli altri partecipanti.

6. Cronologia delle partite

- Come utente, voglio poter consultare lo storico delle mie partite precedenti, in modo da rivedere le mie performance e migliorare.
- Come utente, voglio rivedere i dettagli delle mie partite passate contro l'agente intelligente per analizzare le mie strategie.

7. Interfaccia di gioco responsive

- Come utente, voglio un'interfaccia di gioco che si adatti a vari dispositivi (desktop, tablet, mobile), così posso giocare ovunque e in qualsiasi momento.
- Come utente, voglio un'interfaccia chiara e intuitiva per eseguire mosse durante il gioco, in modo che il gameplay sia fluido e piacevole.

8. Notifiche e avvisi

- Come utente, voglio ricevere notifiche quando è il mio turno in una partita o in un torneo, così non perdo i miei turni.
- Come utente, voglio ricevere notifiche di aggiornamento del mio rating o del posizionamento nella classifica, così posso monitorare i miei progressi.

Queste user stories coprono le principali esigenze funzionali del prodotto e mettono l'accento sull'esperienza utente, sulle interazioni sociali e sul gameplay.

PO

Esempi di epics per queste user stories?

ChatGPT

Gli **epics** sono funzionalità di alto livello che possono essere suddivise in varie **user stories**. Per il prodotto **Backgammon social site**, gli **epics** che comprendono le user stories menzionate potrebbero essere i seguenti:

1. Gestione account utente e autenticazione

- **Descrizione:** Implementare il sistema di registrazione e autenticazione per gli utenti del sito.
- **User stories correlate:**
 - Come utente, voglio potermi registrare e accedere al sito con il mio account personale.
 - Come utente, voglio poter collegare il mio account social (Google o Facebook) per accedere più velocemente.

2. Gestione delle partite di backgammon

- **Descrizione:** Permettere agli utenti di sfidare altri giocatori o un agente intelligente e di giocare partite di backgammon online.
- **User stories correlate:**
 - Come utente, voglio poter sfidare un'altra persona online a una partita di backgammon.
 - Come utente, voglio poter giocare contro un agente intelligente (AI) per allenarmi.

3. Organizzazione e gestione tornei

- **Descrizione:** Creare un sistema che consenta agli utenti di organizzare e partecipare a tornei di backgammon con un massimo di 4 giocatori.
- **User stories correlate:**

- Come utente, voglio poter organizzare tornei con un massimo di 4 giocatori.
- Come utente, voglio visualizzare il calendario e il tabellone dei tornei.

4. Leaderboard e sistema di rating

- **Descrizione:** Implementare un sistema di classifica e rating basato sulle prestazioni dei giocatori.
- **User stories correlate:**
 - Come utente, voglio vedere la mia posizione nella classifica globale basata sul mio rating personale.
 - Come utente, voglio che il mio rating sia aggiornato automaticamente alla fine di ogni partita o torneo.

5. Comunicazione tra giocatori

- **Descrizione:** Consentire agli utenti di comunicare tra di loro durante il gioco e nei tornei.
- **User stories correlate:**
 - Come utente, voglio poter inviare messaggi al mio avversario durante la partita.
 - Come utente, voglio avere accesso a una chat di gruppo per i tornei.

6. Storico delle partite e statistiche

- **Descrizione:** Offrire agli utenti la possibilità di visualizzare la cronologia delle loro partite e delle loro prestazioni.
- **User stories correlate:**
 - Come utente, voglio poter consultare lo storico delle mie partite precedenti.
 - Come utente, voglio rivedere i dettagli delle mie partite passate contro l'agente intelligente.

7. Interfaccia di gioco responsive

- **Descrizione:** Progettare un'interfaccia di gioco che funzioni su diversi dispositivi e offra un'esperienza utente fluida.
- **User stories correlate:**
 - Come utente, voglio un'interfaccia di gioco che si adatti a vari dispositivi (desktop, tablet, mobile).
 - Come utente, voglio un'interfaccia chiara e intuitiva per eseguire mosse durante il gioco.

8. Sistema di notifiche e avvisi

- **Descrizione:** Implementare un sistema di notifiche che avverte gli utenti su turni di gioco, tornei e aggiornamenti di classifica.
- **User stories correlate:**
 - Come utente, voglio ricevere notifiche quando è il mio turno in una partita o in un torneo.
 - Come utente, voglio ricevere notifiche di aggiornamento del mio rating o del posizionamento nella classifica.

Ogni **epic** può essere suddiviso ulteriormente in **user stories**, che a loro volta possono essere scomposte in **task** tecnici per la realizzazione del prodotto.

A.2 Supporto all'analisi della fattibilità

Umano

Abbiamo scelto il progetto 3. Ti ho allegato l'analisi dei rischi già fatta. Crea l'analisi della fattibilità.
[Allegati il pdf di descrizione del progetto e quello dell'analisi dei rischi]

ChatGPT

1. Fattibilità Tecnica

- **Tecnologie richieste:**

- Il progetto richiede lo sviluppo di un gioco multiplayer, l'integrazione con i social network e l'utilizzo di container Docker per il rilascio del prodotto. Tecnologie come WebSocket per il multiplayer e API per i social network saranno necessarie.
- Gli strumenti come GitLab, SonarQube e Jenkins supportano un processo di sviluppo continuo e sicuro, mentre l'uso di Docker per il deploy garantisce la portabilità e facilità di distribuzione.

- **Rischi tecnici principali:**

- **Sviluppo AI:** L'intelligenza artificiale (AI) è una delle parti più complesse, soprattutto per garantire decisioni rapide e sensate nel contesto di gioco (Backgammon). Tuttavia, l'uso di librerie esistenti e l'implementazione graduale (AI semplificata inizialmente) riduce i rischi.

2. Fattibilità Economica

- **Costi di sviluppo:**

- Il progetto fa ampio uso di strumenti open source come Docker, GitLab, SonarQube, Taiga, che riducono i costi di sviluppo. Tuttavia, il costo legato ai server privati per l'hosting potrebbe essere significativo, soprattutto considerando le esigenze di prestazioni elevate per il multiplayer e il salvataggio dei dati.

- **Costi aggiuntivi:**

- Oltre ai costi dei server, ci potrebbero essere costi per garantire la sicurezza delle connessioni (crittografia, protocolli SSL/TLS) e per mantenere il software aggiornato in caso di modifiche alle API dei social network.

3. Fattibilità Tecnica

- **Componenti critiche:**

- **Intelligenza Artificiale:** Lo sviluppo di un'AI competitiva richiede competenze avanzate, ma l'uso di algoritmi noti (come il minimax per i giochi da tavolo) e librerie open source riduce il rischio tecnico. Iniziare con una versione semplificata dell'AI e poi migliorarla progressivamente permette di distribuire il carico tecnico.
- **Multiplayer:** La gestione del multiplayer in tempo reale presenta sfide significative, ma l'uso di WebSocket per la sincronizzazione in tempo reale è una soluzione tecnicamente fattibile. Test intensivi e una gestione ottimale delle disconnessioni sono fondamentali per evitare problemi di esperienza utente.
- **Sicurezza:** Il gioco online deve affrontare rischi di sicurezza, soprattutto legati ai dati degli utenti e alle connessioni non sicure. Strumenti come SonarQube aiutano a rilevare vulnerabilità, mentre l'adozione di protocolli di sicurezza come SSL/TLS è essenziale.

4. Fattibilità Temporale

- **Durata prevista:**

- Il progetto prevede almeno tre sprint di due settimane ciascuno, con un prototipo funzionante previsto entro la fine del corso. Il rispetto delle scadenze dipende dalla suddivisione del lavoro in moduli incrementali e dal rilascio progressivo di versioni minime funzionanti (MVP).

- **Criticità temporali:**

- Le funzionalità più complesse come AI e multiplayer potrebbero richiedere più tempo del previsto, soprattutto se il team non ha familiarità con queste tecnologie. Suddividere lo sviluppo in componenti più piccoli e rilasciare versioni preliminari aiuterà a gestire i tempi di sviluppo.

5. Fattibilità Legale

- **Conformità alle normative sulla privacy:**

- Il progetto richiede il rispetto delle normative sulla protezione dei dati personali (come il GDPR in Europa), poiché i dati dei giocatori verranno raccolti e gestiti sui server privati. Sarà necessario implementare misure per garantire la privacy e la sicurezza delle informazioni degli utenti.

- **Uso di API esterne:**

- L'integrazione con le API dei social network potrebbe comportare vincoli legali legati ai termini d'uso delle piattaforme stesse (Facebook, Twitter, Instagram). È importante garantire che l'integrazione sia conforme alle politiche delle piattaforme, soprattutto per quanto riguarda la condivisione di dati e la gestione degli account utente.

- **Licenze open source:**

- L'uso di librerie e codice open source richiede attenzione per quanto riguarda le licenze. Bisogna assicurarsi che le licenze delle librerie utilizzate siano compatibili con il progetto e che non ci siano restrizioni che potrebbero influenzare la distribuzione del prodotto finale.

Conclusioni Generali

Il progetto è fattibile sotto tutti i principali aspetti analizzati:

- **Le risorse necessarie:** Le risorse tecnologiche e umane sono accessibili, ma sarà necessaria formazione in alcune aree chiave.
- **Fattibilità economica:** I costi possono essere contenuti grazie all'uso di strumenti open source, ma la gestione dei server e della sicurezza richiede un investimento iniziale che deve essere pianificato con attenzione.
- **Fattibilità tecnica:** Il progetto è tecnicamente fattibile, con alcune criticità legate alla complessità dell'AI e del multiplayer, ma gestibili tramite un approccio incrementale.
- **Fattibilità temporale:** Le tempistiche possono essere rispettate se le funzionalità complesse vengono pianificate e sviluppate in modo modulare.
- **Fattibilità legale:** Il progetto è fattibile dal punto di vista legale, a condizione che vengano rispettate le normative sulla privacy, le licenze open source e i termini d'uso delle API dei social network.

In sintesi, il progetto è fattibile, ma richiede un'attenzione particolare alla gestione delle risorse, alla pianificazione del lavoro e alla conformità legale.

Umano

Rifai l'analisi considerando questi aspetti: disponibilità delle risorse necessarie, fattibilità tecnica, rispetto delle tempistiche, fattibilità economica e fattibilità organizzativa.

ChatGPT

1. Disponibilità delle Risorse Necessarie

- **Risorse umane:**

- Il team deve essere composto da almeno cinque persone, ciascuna con un ruolo definito (Product Owner, Scrum Master, Developer).

- È necessario verificare la disponibilità di membri del team con competenze specifiche nelle aree chiave (sviluppo AI, gestione del multiplayer, container Docker).
- La formazione interna per colmare eventuali lacune nelle competenze (AI e multiplayer) può richiedere tempo, ma è fattibile grazie a risorse open source e sessioni di autoapprendimento.

- **Infrastruttura e strumenti:**

- Il progetto fa affidamento su tecnologie open source come Docker, GitLab, SonarQube, strumenti ampiamente disponibili e consolidati. Tuttavia, la necessità di server privati per ospitare il gioco e i dati dei giocatori richiede risorse aggiuntive.
- La disponibilità di questi server e la loro configurazione sono fattori critici per la continuità del servizio multiplayer.

2. Fattibilità Economica

- **Costi di sviluppo:**

- L'utilizzo di strumenti open source (Docker, GitLab, SonarQube, Taiga) riduce significativamente i costi di sviluppo software. Tuttavia, il costo legato ai server privati per l'hosting potrebbe essere significativo, soprattutto considerando le esigenze di prestazioni elevate per il multiplayer e il salvataggio dei dati.

- **Costi aggiuntivi:**

- Oltre ai costi dei server, ci potrebbero essere costi per garantire la sicurezza delle connessioni (crittografia, protocolli SSL/TLS) e per mantenere il software aggiornato in caso di modifiche alle API dei social network.

3. Fattibilità Tecnica

- **Componenti critiche:**

- **Intelligenza Artificiale:** La creazione di un'intelligenza artificiale per competere contro i giocatori nel contesto di giochi da tavolo come Backgammon è complessa. Tuttavia, l'uso di algoritmi già esistenti (ad esempio, minimax) e librerie open source riduce il rischio tecnico. Iniziare con una versione semplificata dell'AI e poi migliorarla progressivamente permette di distribuire il carico tecnico.
- **Multiplayer:** La gestione del multiplayer in tempo reale richiede una solida infrastruttura di rete. L'uso di WebSocket per una comunicazione in tempo reale a bassa latenza è tecnicamente fattibile, ma richiede test intensivi per prevenire problemi di disconnessione o lag.
- **Sicurezza:** Il gioco online deve affrontare rischi di sicurezza, soprattutto legati ai dati degli utenti e alle connessioni non sicure. Strumenti come SonarQube aiutano a rilevare vulnerabilità, mentre l'adozione di protocolli di sicurezza come SSL/TLS è essenziale.

4. Rispetto delle Tempistiche

- **Durata prevista:**

- Il progetto si svolgerà attraverso almeno tre sprint di due settimane ciascuno, con un prototipo funzionante previsto entro la fine del corso. Le tempistiche dipendono dalla suddivisione del lavoro in moduli incrementali e dal rilascio progressivo di versioni minime funzionanti (MVP).

- **Criticità temporali:**

- Le funzionalità più complesse, come l'AI e il multiplayer, potrebbero richiedere più tempo del previsto. Tuttavia, suddividere lo sviluppo in componenti più piccoli e rilasciare versioni preliminari aiuterà a gestire i tempi di sviluppo.

5. Fattibilità Organizzativa

- **Team e Ruoli:**

- La divisione dei ruoli tra Product Owner, Scrum Master e Developer garantisce una gestione efficace del progetto. Il processo Scrum permette una revisione continua del progresso e una gestione flessibile delle problematiche emergenti.

- **Gestione del rischio:**

- I rischi tecnici e operativi sono stati ben identificati nell'analisi dei rischi, e sono presenti strategie di mitigazione che consentono di affrontare i problemi in modo proattivo. Pianificare revisioni periodiche con il cliente (Requirement Owner) e mantenere un backlog costantemente aggiornato aiuterà a prevenire errori e ritardi.

Conclusioni Generali

Il progetto è fattibile sotto tutti i principali aspetti analizzati:

- **Le risorse necessarie:** Le risorse tecnologiche e umane sono accessibili, ma sarà necessaria formazione in alcune aree chiave.
- **Fattibilità economica:** I costi possono essere contenuti grazie all'uso di strumenti open source, ma la gestione dei server e della sicurezza richiede un investimento iniziale che deve essere pianificato con attenzione.
- **Fattibilità tecnica:** Il progetto è tecnicamente fattibile, con alcune criticità legate alla complessità dell'AI e del multiplayer, ma gestibili tramite un approccio incrementale.
- **Rispetto delle tempistiche:** Le tempistiche possono essere rispettate se le funzionalità complesse vengono pianificate e sviluppate in modo modulare.
- **Fattibilità organizzativa:** L'organizzazione interna del team e l'uso del framework Scrum garantiscono un'adeguata gestione del progetto, con la possibilità di adattarsi a eventuali problemi tecnici e organizzativi.

In sintesi, il progetto è fattibile, ma richiede un'attenzione particolare alla gestione delle risorse, alla pianificazione del lavoro e alla conformità legale.

A.3 Supporto alla scrittura di test

A.3.1 Esempio con ChatGPT

Developer

write the vitest test for this function:

```
export const findUsedDie = (dice: number[],  
srcPointIndex: number, dstPointIndex: number) => {  
  const usedDice = dice.reduce((min, die) => {  
    if (srcPointIndex - die <= dstPointIndex && die < min) {  
      return die;  
    }  
    return min;  
  }, Infinity);  
  
  if (usedDice === Infinity || usedDice === undefined) {  
    return null;  
  } else {  
    return usedDice;  
  }  
};
```

ChatGPT

To write a test for the `findUsedDie` function using Vitest, we need to focus on testing the behavior of the function with various inputs. The function finds the smallest die that satisfies the condition `srcPointIndex - die <= dstPointIndex` and returns it, or returns null if no such die is found.

Here's how you can write Vitest tests for this function:

1. **Basic Test:** Testing when a valid die is found.
2. **Edge Case:** Testing when no die satisfies the condition.
3. **Empty Dice Array:** Testing when the dice array is empty.

```
import { describe, it, expect } from 'vitest';
import { findUsedDie } from './path-to-your-function-file';

describe('findUsedDie', () => {
  it('should return the correct die used for the move', () => {
    const dice = [1, 2, 3, 4, 5]
    const srcPointIndex = 10
    const dstPointIndex = 7

    const usedDie = findUsedDie(dice, srcPointIndex, dstPointIndex)

    expect(usedDie).toBe(3)
  })

  it('should return null if no die can be used for the move', () => {
    const dice = [1, 2, 3, 4, 5]
    const srcPointIndex = 10
    const dstPointIndex = 4

    const usedDie = findUsedDie(dice, srcPointIndex, dstPointIndex)

    expect(usedDie).toBeNull()
  })

  it('should return null if dice array is empty', () => {
    const dice: number[] = []
    const srcPointIndex = 10
    const dstPointIndex = 9

    const usedDie = findUsedDie(dice, srcPointIndex, dstPointIndex)

    expect(usedDie).toBeNull()
  })
})
```

A.3.2 Esempio con Copilot

Si noti che Copilot tendeva a mescolare Jest e Vitest nonostante le richieste specificassero Vitest, rendendo necessario un successivo adattamento.

Developer

How do i test this with vitest?

Copilot

To test a Vue component with Vitest, you will need to set up Vitest in your project and create test files for your components. Here are the steps to get you started:

1. **Install Vitest and Vue Test Utils:** Install the necessary packages to work with Vitest and

Vue Test Utils by running:

```
npm install --save-dev vitest @vue/test-utils
```

2. **Configure Vitest:** Create a configuration file for Vitest if you don't have one already. This could be `vite.config.ts` or `vitest.config.ts` depending on your setup.
3. **Create a Test File:** Create a test file for your `TournamentLists.vue` component. For example, you can create `TournamentLists.test.ts` in the same directory or in a `tests` directory.
4. **Write Tests:** Use Vue Test Utils and Vitest to write your tests. Here is an example of how you might write a test for your Vue component:

```
import { mount } from '@vue/test-utils'
import { describe, it, expect } from 'vitest'
import TournamentLists from './TournamentLists.vue'

describe('TournamentLists.vue', () => {
  it('renders correctly', () => {
    const wrapper = mount(TournamentLists)
    expect(wrapper.exists()).toBeTruthy()
  })

  // Add more tests as needed
})
```

5. **Run the Tests:** Add a script to your `package.json` to run the tests:

```
"scripts": {
  "test": "vitest"
}
```

Then run the tests using:

```
npm run test
```

This is a basic setup to get you started with testing your Vue component using Vitest. You can add more tests to cover the functionality of your component as needed.

Developer

Add more tests

Copilot

It looks like there isn't a test file for `TournamentLists.vue` yet. Let's create one and add tests for the component. Here is an example test file using Vitest and Vue Test Utils:

```
// client/src/components/TournamentLists.test.ts
import { mount } from '@vue/test-utils'
import { describe, it, expect } from 'vitest'
import TournamentLists from './TournamentLists.vue'

describe('TournamentLists.vue', () => {
  it('renders correctly', () => {
    const wrapper = mount(TournamentLists)
    expect(wrapper.exists()).toBeTruthy()
  })
})
```

```

it('displays the correct headings', () => {
  const wrapper = mount(TournamentLists)
  expect(wrapper.find('h3').text()).toBe('OPEN')
  expect(wrapper.findAll('h3')[1].text()).toBe('CLOSED')
})

it('displays the correct message when there are no tournaments', () => {
  const wrapper = mount(TournamentLists, {
    data() {
      return {
        openTournaments: [],
        closedTournaments: []
      }
    }
  })
  expect(wrapper.find('.text-gray-500').text())
    .toBe('No open tournaments available.')
  expect(wrapper.findAll('.text-gray-500')[1].text())
    .toBe('No closed tournaments available.')
})

it('calls selectTournament method when a button is clicked', async () => {
  const selectTournament = jest.fn()
  const wrapper = mount(TournamentLists, {
    methods: { selectTournament }
  })
  await wrapper.find('button').trigger('click')
  expect(selectTournament).toHaveBeenCalled()
})
})

```

Add this file to your project in the `client/src/components` directory. Then, run your tests with the following command:

`npm run test`

This will set up your initial tests for the `TournamentLists.vue` component. You can add more tests as needed to cover additional functionality.

A.4 Supporto all'inizializzazione di nuove funzionalità

Developer

Add a password recovery button and implement its logic

Copilot

To add a password recovery button and implement its logic, follow these steps:

1. Add a button for password recovery in the template.
2. Create a method to handle the password recovery process.
3. Update the `registerUser` method to handle the password recovery request.

Here is the updated code:

Update the template to add the password recovery button

```
<template>
  <!-- Existing template code -->
```

```

<div v-if="!showRegisterForm" class="flex flex-wrap items-center
justify-between gap-4 mt-6">
<div>
  <a @click.prevent="recoverPassword" class="text-green-600 font-semibold
text-sm hover:underline">
    Forgot Password?
  </a>
</div>
</div>
<!-- Existing template code -->
</template>

```

Add the password recovery method in the script section

```

<script lang="ts">
import { defineComponent, ref } from 'vue';
import axios from 'axios';
import { useRouter } from 'vue-router';
import { registerOrLogin, recoverPassword } from '@/services/authService';

export default defineComponent({
  name: 'RegisterView',
  setup() {
    const router = useRouter();
    const showRegisterForm = ref(false);
    const showPassword = ref(false);
    const name = ref('');
    [Continue]

```

Continuation of the test_game.py file with additional test cases

```

@ pytest.mark.anyio
async def test_end_game(client: AsyncClient, token: str):
    await clear_matches()
    await create_started_match("testuser", "testuser2")
    await get_db().matches.update_one({"player1": "testuser"}, {
        "$set": {"turn": 20, "dice": [3, 5], "available": [3, 5],
                  "rounds_to_win": 1, "winsP1": 0, "winsP2": 0}})
    move_data = {
        "board": {
            "points": [{"player1": 1, "player2": 0}] + [{"player1": 0, "player2": 0}
                for _ in range(23)],
            "bar": {"player1": 0, "player2": 0}
        }
    }
    await client.post("/move/piece", json=move_data,
                      headers={"Authorization": f"Bearer {token}"})
    updated_game = await get_db().matches.find_one({"player1": "testuser"})
    assert updated_game is not None
    assert updated_game["turn"] == -1
    assert updated_game["winsP1"] == 1
    assert updated_game["game_over"] is True

@ pytest.mark.anyio
async def test_invalid_move(client: AsyncClient, token: str):
    await clear_matches()
    await create_started_match("testuser", "testuser2")
    await get_db().matches.update_one({"player1": "testuser"}, {
        "$set": {"dice": [3, 5], "available": [3, 5]}})
```

```

move_data = {
    "board": {
        "points": [{"player1": 0, "player2": 1} for _ in range(24)],
        "bar": {"player1": 0, "player2": 0}
    },
    "dice": 3
}
response = await client.post("/move/piece", json=move_data,
    headers={"Authorization": f"Bearer {token}"})
assert response.status_code == 400
assert response.json()["detail"] == "Invalid move"

```

These additional test cases cover scenarios such as ending the game and handling invalid moves.

Developer

Add password recovery logic

Copilot

To add password recovery logic, you need to implement both the frontend and backend parts. Here is a brief overview of the steps:

- Backend:** Implement an endpoint to handle password recovery requests and another to reset the password.
- Frontend:** Add a form to request a password reset and another to set a new password.

Backend

1. Password Recovery Request Endpoint:

- This endpoint will accept an email address and send a password recovery email with a unique token.

2. Password Reset Endpoint:

- This endpoint will accept the token and the new password to reset the user's password.

Frontend

1. Password Recovery Form:

- A form where users can enter their email to request a password reset.

2. Password Reset Form:

- A form where users can enter a new password using the token they received in their email.

Backend Code (Python)

Add the following endpoints to your backend:

```

from fastapi import APIRouter, HTTPException, Depends
from pydantic import BaseModel
from fastapi.security import OAuth2PasswordRequestForm
from some_email_service import send_password_reset_email
from some_database_service import get_user_by_email, update_user_password
from some_token_service import create_reset_token, verify_reset_token

router = APIRouter()

class PasswordRecoveryRequest(BaseModel):

```

```

email: str

class PasswordResetRequest(BaseModel):
    token: str
    new_password: str

@router.post("/password-recovery")
async def password_recovery(request: PasswordRecoveryRequest):
    user = await get_user_by_email(request.email)
    if not user:
        raise HTTPException(status_code=404, detail="User not found")
    token = create_reset_token(user.id)
    await send_password_reset_email(user.email, token)
    return {"message": "Password recovery email sent"}


@router.post("/password-reset")
async def password_reset(request: PasswordResetRequest):
    user_id = verify_reset_token(request.token)
    if not user_id:
        raise HTTPException(status_code=400, detail="Invalid or expired token")
    await update_user_password(user_id, request.new_password)
    return {"message": "Password has been reset"}

```

Frontend Code (Vue)

Add the following components to your Vue application:

PasswordRecoveryView.vue

```

<template>
  <div>
    <h3>Password Recovery</h3>
    <form @submit.prevent="requestPasswordRecovery">
      <input v-model="email" type="email" placeholder="Enter your email" required />
      <button type="submit">Send Recovery Email</button>
    </form>
    <p v-if="message">{{ message }}</p>
  </div>
</template>

<script lang="ts">
import { defineComponent, ref } from 'vue';
import axios from 'axios';

export default defineComponent({
  name: 'PasswordRecoveryView',
  setup() {
    const email = ref('');
    const message = ref('');

    const requestPasswordRecovery = async () => {
      try {
        const response = await axios.post('/password-recovery',
          { email: email.value });
        message.value = response.data.message;
      } catch (error) {
        message.value = 'Error sending recovery email';
      }
    };
  }
});

```

```

        return { email, message, requestPasswordRecovery };
    },
});
</script>

```

PasswordResetView.vue

```

<template>
  <div>
    <h3>Reset Password</h3>
    <form @submit.prevent="resetPassword">
      <input v-model="token" type="text" placeholder="Enter your token" required />
      <input v-model="newPassword" type="password"
        placeholder="Enter new password" required />
      <button type="submit">Reset Password</button>
    </form>
    <p v-if="message">{{ message }}</p>
  </div>
</template>

<script lang="ts">
import { defineComponent, ref } from 'vue';
import axios from 'axios';

export default defineComponent({
  name: 'PasswordResetView',
  setup() {
    const token = ref('');
    const newPassword = ref('');
    const message = ref('');

    const resetPassword = async () => {
      try {
        const response = await axios.post('/password-reset',
          { token: token.value, new_password: newPassword.value });
        message.value = response.data.message;
      } catch (error) {
        message.value = 'Error resetting password';
      }
    };

    return { token, newPassword, message, resetPassword };
  },
});
</script>

```

These components will allow users to request a password reset and set a new password using the token they receive via email.