VezGammon - Report finale Team 1

Progetto di Ingegneria del Software

Lorenzo Peronese - Scrum Master

22 gennaio 2025

Indice

1 Descrizione del prodotto

1.1 Introduzione

Per il progetto del corso di Ingegneria del Software, il team ha scelto di lavorare a VezGammon™, un'applicazione web di backgammon. Il progetto ha diversi riferimenti alla città di Bologna, in cui è nato e si è sviluppato: in primis ovviamente il nome (*"vez"* viene usato a Bologna come intercalare e significa amico, fratello); inoltre una volta nel sito il cursore diventa un simpatico tortellino e la board utilizza i colori rosso e blu, rappresentativi della città.

L'applicazione offre un'esperienza di gioco completa con diverse modalità: gli utenti possono sfidarsi in partite locali sullo stesso dispositivo, mettere alla prova le proprie abilità contro intelligenze artificiali di vari livelli di difficoltà e confrontarsi online con altri giocatori attraverso un sistema di matchmaking o mediante link di invito diretti. È inoltre possibile organizzare tornei a quattro partecipanti, combinando liberamente giocatori reali e agenti intelligenti.

Il sistema di classificazione si basa sul metodo Elo, ampiamente utilizzato negli scacchi. Ogni nuovo account parte da una base di 800 punti, che vengono poi aggiornati dopo ogni partita online o torneo. Il calcolo dell'aggiornamento tiene conto di diversi fattori: l'esito della partita, l'utilizzo del dado double e il punteggio Elo di entrambi i giocatori. Questo sistema garantisce una competizione equilibrata e permette di mantenere una classifica globale, dove ogni giocatore può aspirare a raggiungere le posizioni più alte.

VezGammon™ include anche ricche funzionalità social e di progressione. Gli utenti possono consultare (e condividere sui social) in ogni momento statistiche dettagliate delle proprie partite, visualizzare grafici che mostrano l'andamento del proprio Elo nel tempo e analizzare i match recenti. Il sistema premia inoltre i giocatori con speciali badge al raggiungimento di specifici obiettivi, come vincere un determinato numero di partite o raggiungere certi livelli di punteggio Elo, aggiungendo un ulteriore elemento di coinvolgimento e progressione al gioco.

1.2 Backlog

NAME	SPRINT	STATUS
#3 come utente, voglio giocare single player contro il computer		Done
#100 come giocatore, voglio poter giocare una partita contro dei bot di diversa difficoltà	sprint2	Done
#26 come creatore di tornei, voglio essere in grado di creare, amministrare tornei	sprint4	Done
#16 come utente, voglio giocare contro altri utenti		Done
#98 come giocatore, voglio poter giocare una partita in locale	sprint2	Done
#99 come giocatore, voglio poter giocare una partita online	sprint3	Done
#26 come creatore di tornei, voglio essere in grado di creare, amministrare tornei	sprint4	Done
#185 come utente, voglio poter personalizzare la mia esperienza di gioco		Done
#24 come giocatore, voglio avere un'interfaccia front-end per navigare all'interno del sito	sprint2	Done
#28 come giocatore, voglio poter scegliere vari temi grafici e ambientazioni	sprint3	Done
#164 come utente, vorrei poter usare il sito anche da cellulare	sprint4	Done
#38 come studente, voglio poter accedere a risorse di addestramento	sprint3	Done
#186 comunicazione e interazion con altri utenti		Done
#37 come giocatore, socievole voglio poter comunicare con i miei avversari in chat	sprint3	Done
#99 come giocatore, voglio poter giocare una partita online	sprint3	Done
#39 come influencer, voglio essere in grado di condividere i miei progressi sui social media	sprint3	Done
#40 come utente, voglio ricevere le notifiche dal sito	sprint4	Done
#187 analisi delle performance		Done
#33 come giocatore, voglio ricevere dei badge digitali come ricompense	sprint3	Done
#34 come studente, voglio essere in grado di analizzare le partite giocate	sprint3	Done
#25 come utente, voglio tener traccia dei miei progressi nel gioco	sprint3	Done
#18 come utente, voglio poter avere un profilo per tenere traccia delle partite giocate e delle statistiche	sprint1	Done

Figura 1: Backlog di Taiga di Vezgammon™

1.3 UML

1.3.1 Casi d'uso

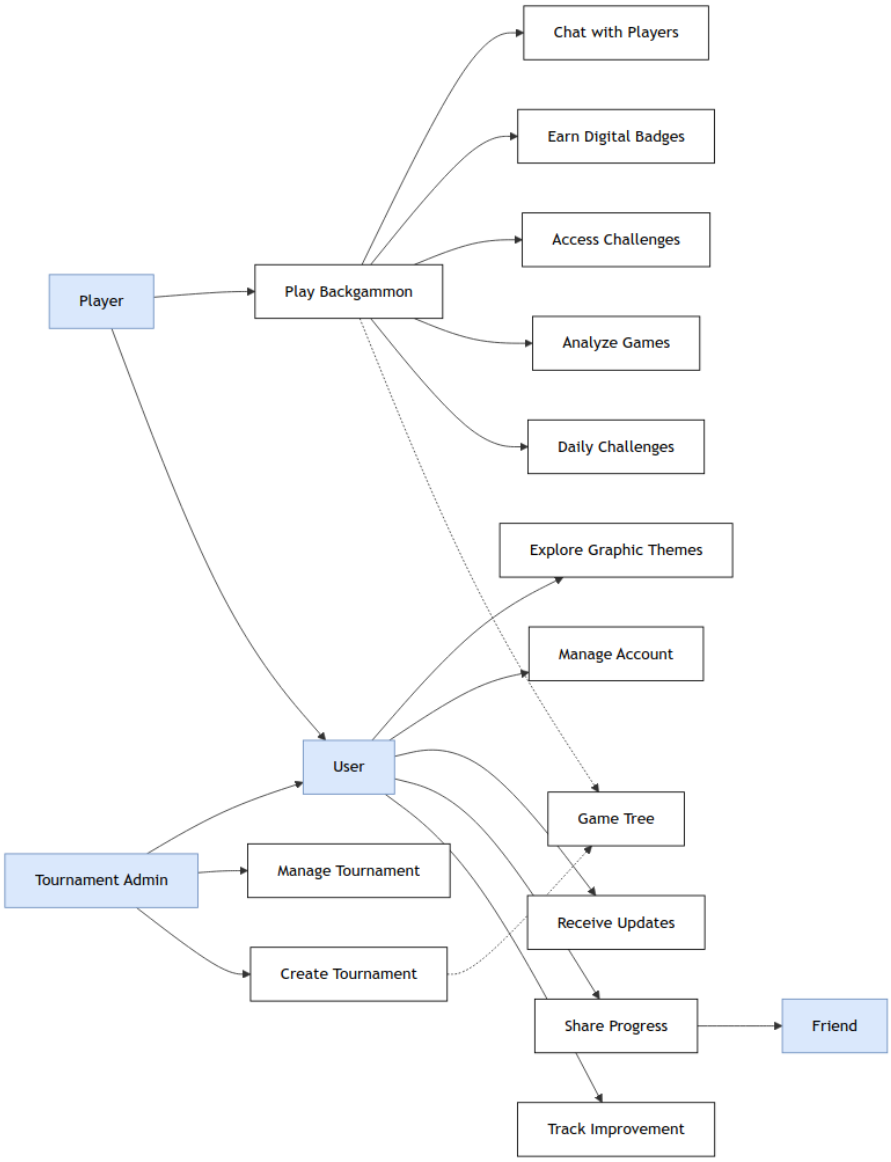

Figura 2: Diagramma UML use-case

1.3.2 Deployment

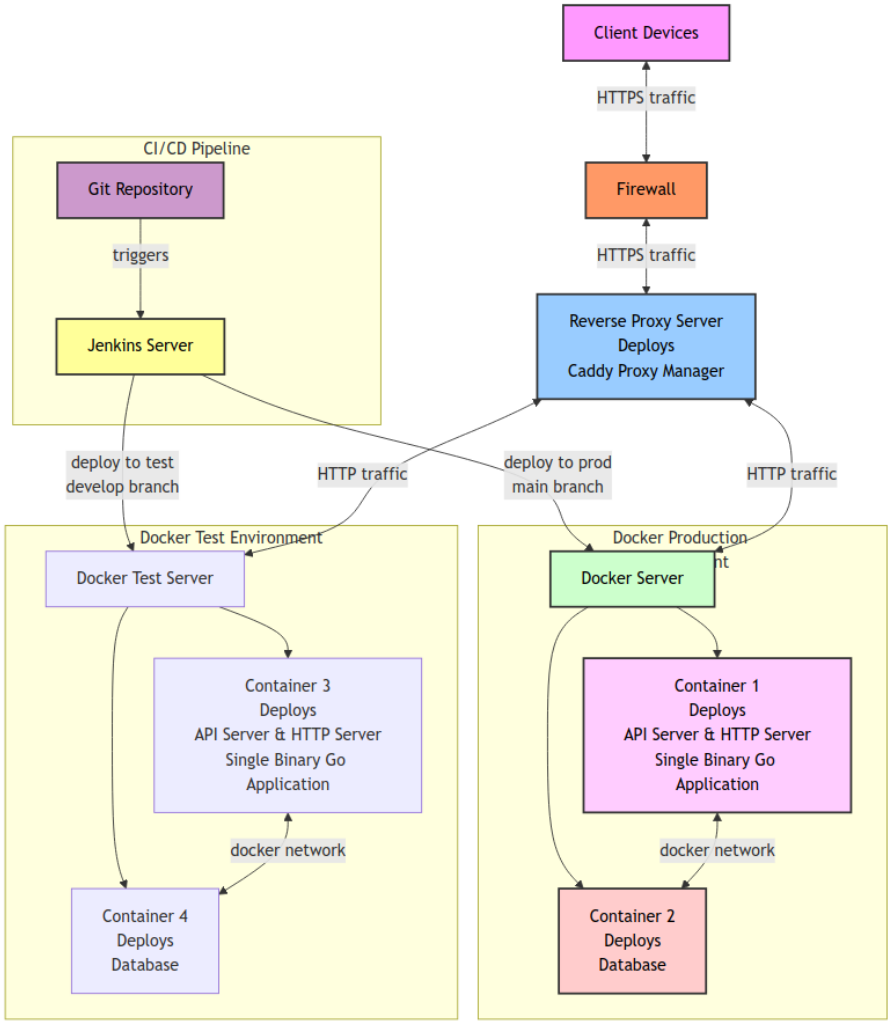

Figura 3: Diagramma UML di deployment

1.3.3 Classi

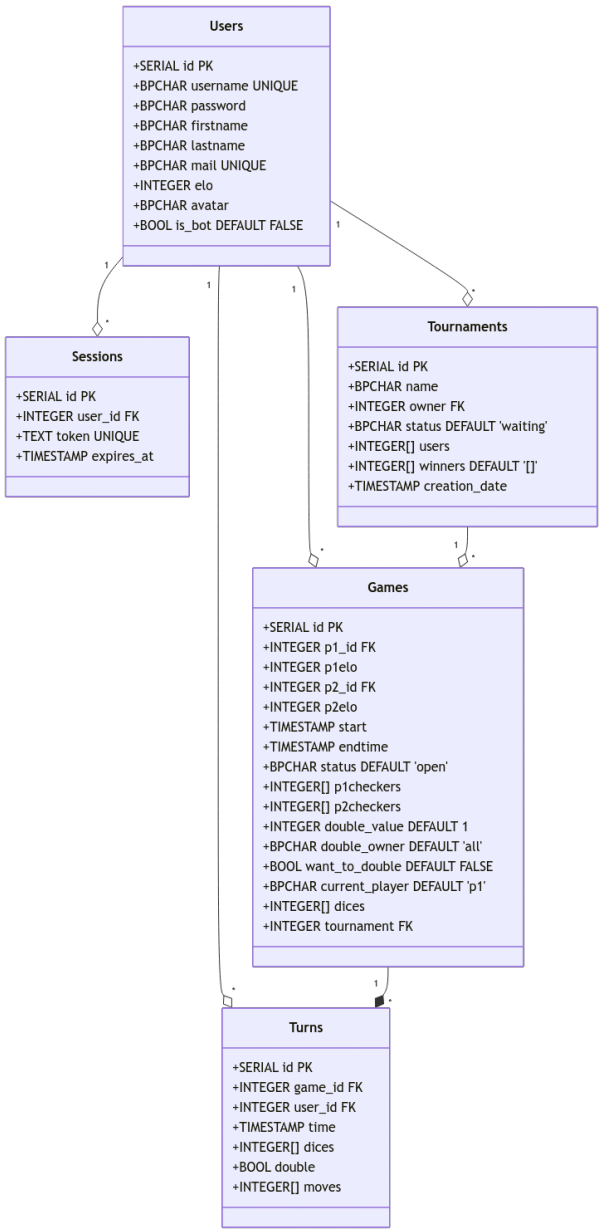

Figura 4: Diagramma UML delle classi

1.3.4 Tornei

Figura 5: Diagramma UML dei tornei

2 Infrastruttura e software utilizzato

2.1 CAS: ambiente di sviluppo agile

Il team ha adottato un approccio completamente open source, basandosi sull'ambiente CAS consigliato dai docenti. Tutti software sono stati self-hostati sotto il dominio `vezgammon.it` per garantire un maggiore controllo e personalizzazionee avere meno dipendenze possibili con l'esterno. Questo ha inizialmente richiesto un notevole sforzo ma a lungo termine sono emersi i benefici di questa scelta.

Le credenziali di accesso sono state fornite agli stakeholder al momento della creazione dell'infrastruttura ma possono essere resettate su richiesta.

2.1.1 GitLab

GitLab è stato utilizzato per la gestione del codice sorgente e per il versioning. Sono stati utilizzati i tag per denotare il Minimun Value Product di ogni sprint. Il team ha qui configurato il repository, gestito le merge request e segnalato e risolto le issue.

Il repository è accessibile all'indirizzo `gitlab.vezgammon.it`

2.1.2 Taiga

Taiga è stato scelto per la gestione del project management. È stato utilizzato per tracciare le attività, gestire il backlog (vedi 1.2), definire le user stories e monitorare lo stato di avanzamento del progetto durante gli sprint.

Il progetto Taiga è accessibile all'indirizzo `taiga.vezgammon.it`

2.1.3 MatterMost

Come descritto nella Sezione 4.5, MatterMost è stato utilizzato per la comunicazione interna del team. È stata la piattaforma principale per gli scambi di messaggi, la condivisione di aggiornamenti e per accordarsi sugli incontri quotidiani.

MatterMost self-hosted è accessibile all'indirizzo `mattermost.vezgammon.it`

2.1.4 Jenkins

Jenkins è stato utilizzato come *core* per l'automazione del flusso di lavoro tramite CI/CD e dei processi di build e deployment. Il team ha configurato job automatici per costruire, testare e distribuire l'applicazione.

Jenkins self-hosted è accessibile all'indirizzo `jenkins.vezgammon.it`

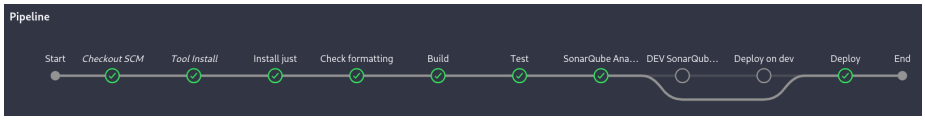

Figura 6: Pipeline di Jenkins: viene lanciata a ogni evento push su GitLab e controlla formattazione, build e test; sui commit delle branch *main* e *develop* viene anche eseguito il controllo statico del codice con SonarQube e il deploy sul relativo server, di produzione o di sviluppo.

2.1.5 SonarQube

Come descritto nella Sezione 4.7, SonarQube è stato utilizzato per l'analisi della qualità del codice. Il team ha configurato SonarQube per eseguire analisi statiche e rilevare eventuali problematiche riguardanti la qualità del codice, la copertura dei test e la manutenzione.

SonarQube self-hosted è accessibile all'indirizzo `sonarqube.vezgammon.it`

2.1.6 Swagger

Per la documentazione delle API implementate e il testing più rapido, il team ha scelto di utilizzare il tool *Swagger*(`https://swagger.io/`).

Vengono di seguito riportate alcune immagini che ne illustrano il funzionamento.

Figura 7: API utilizzate per gestire una generica partita

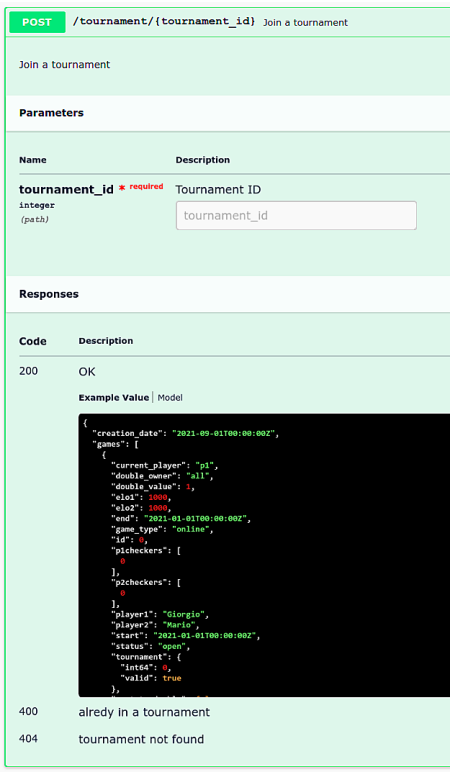

Figura 8: Nel dettaglio, l'interfaccia di Swagger per ogni API

2.1.7 Status

Il sistema di monitoraggio dei servizi è stato sviluppato per tenere traccia della disponibilità dei vari strumenti utilizzati nel progetto.

É stato inoltre sviluppato un bot Telegram per avvisare tempestivamente il team di eventuali problemi con il server e/o i singoli servizi.

Il sito che riassume lo stato dei servizi è accessibile all'indirizzo `status.vezgammon.it`

2.2 Server di deploy

Il team ha configurato due server principali per ospitare l'applicazione:

- **Server di produzione**: `vezgammon.it`
- **Server di sviluppo**: `dev.vezgammon.it`

Il server di produzione è stato utilizzato per la versione stabile, mentre il server di sviluppo è stato dedicato al testing, al debug e alle fasi di sviluppo continuo.

3 Processo Scrum

TODO MANCA SOLO QUESTO

3.1 Sprint 0

Oltre alle attività di teambuilding (vedi 4.2), i membri del gruppo hanno iniziato in questa fase a studiare il funzionamento e la documentazione dei software

3.2 Sprint 1

3.2.1 Goal

3.2.2 Backlog

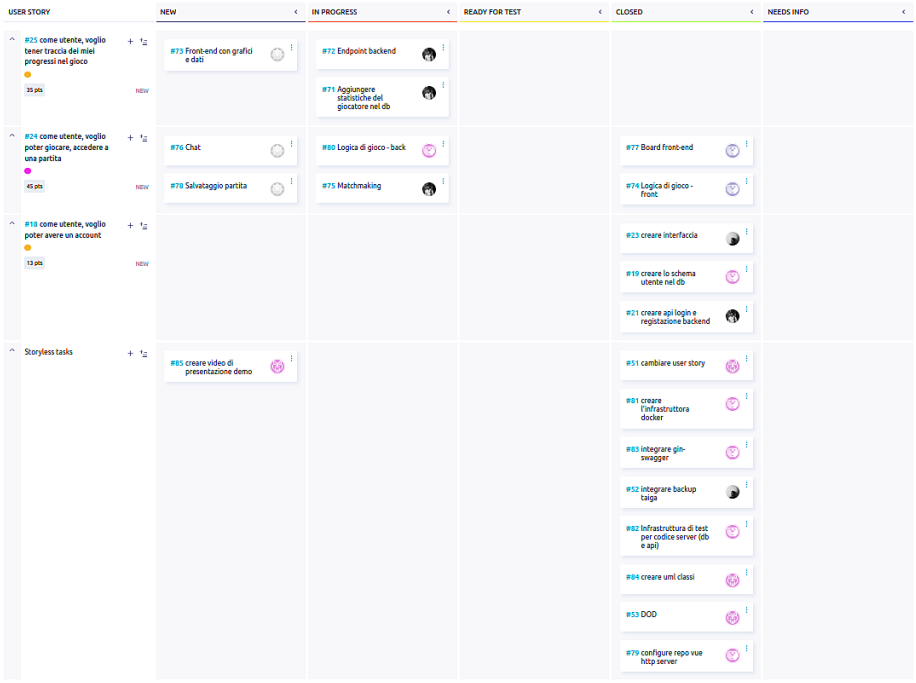

Figura 9: Backlog del primo sprint

3.2.3 Incremento

3.2.4 Definition of Done

3.2.5 Esempio di test fatti

3.2.6 Burndown

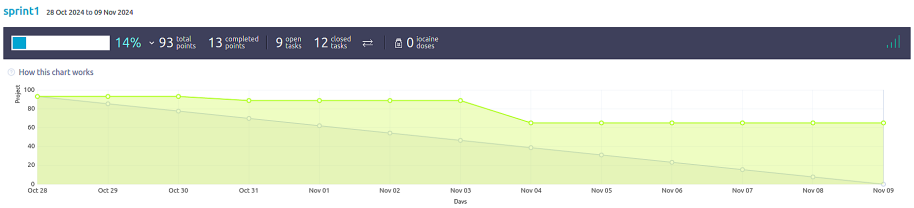

Figura 10: Burndown del primo sprint

3.2.7 Retrospettiva

3.3 Sprint 2

3.3.1 Goal

3.3.2 Backlog

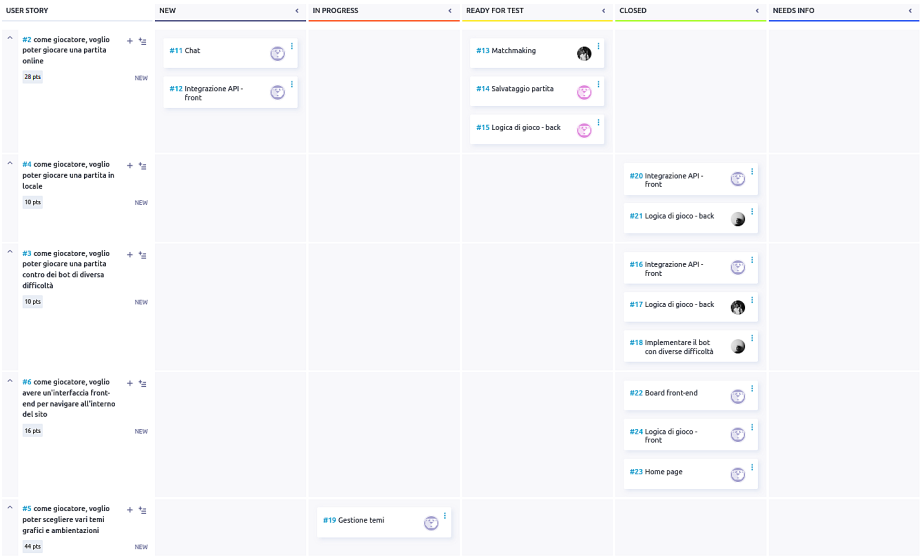

Figura 11: Backlog del secondo sprint - parte 1

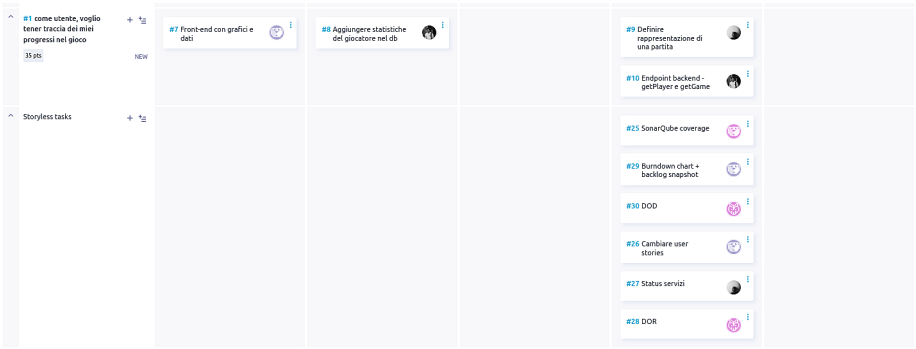

Figura 12: Backlog del secondo sprint - parte 2

Purtroppo l'immagine originale era molto sfocata, quindi ho riprodotto il progetto di Taiga per fare questi screenshots. Per questo motivo i numeri sulle user stories e sui task non coincidono con quelli delle altre immagini.

3.3.3 Incremento

3.3.4 Definition of Done

3.3.5 Esempio di test fatti

3.3.6 Burndown

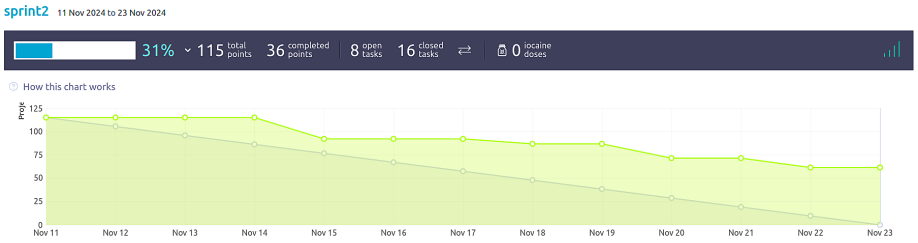

Figura 13: Burndown del secondo sprint

3.3.7 Retrospettiva

3.4 Sprint 3

3.4.1 Goal

3.4.2 Backlog

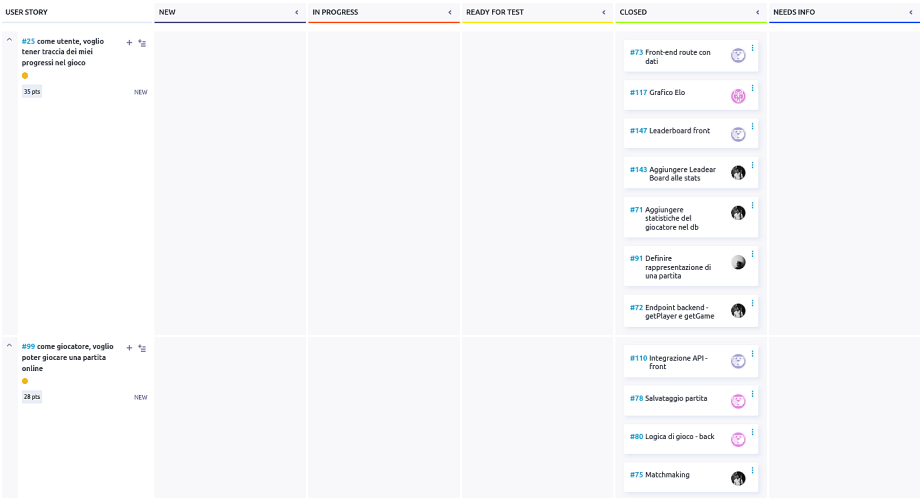

Figura 14: Backlog del terzo sprint - parte 1

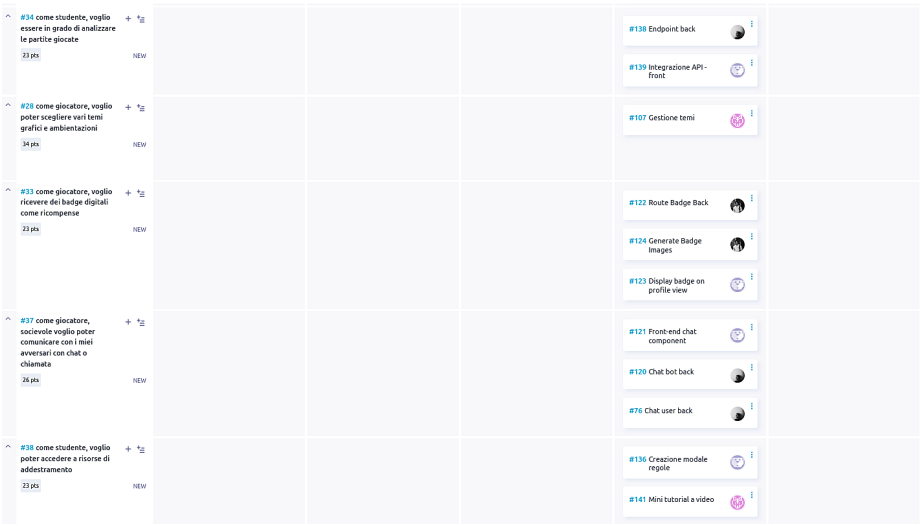

Figura 15: Backlog del terzo sprint - parte 2

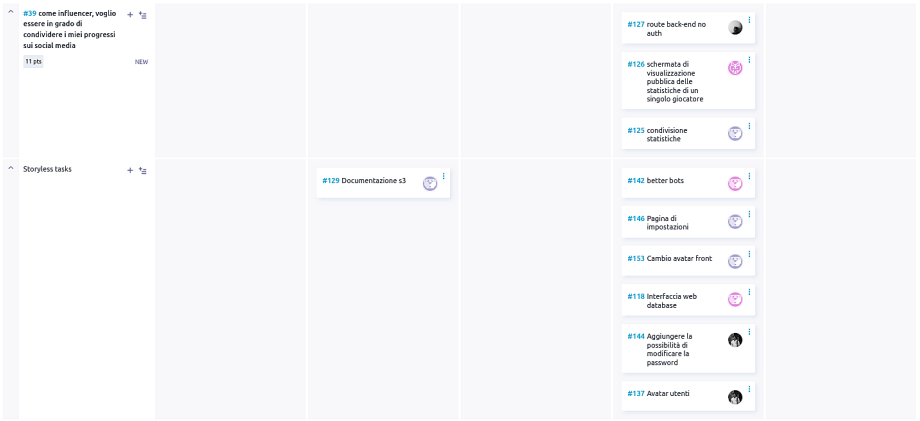

Figura 16: Backlog del terzo sprint - parte 3

3.4.3 Incremento

3.4.4 Definition of Done

3.4.5 Esempio di test fatti

3.4.6 Burndown

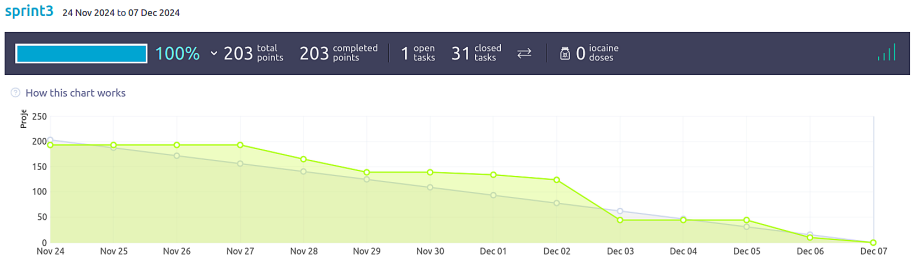

Figura 17: Burndown del terzo sprint

3.4.7 Retrospettiva

3.5 Sprint 4

3.5.1 Goal

3.5.2 Backlog

Figura 18: Backlog del quarto sprint - parte 1

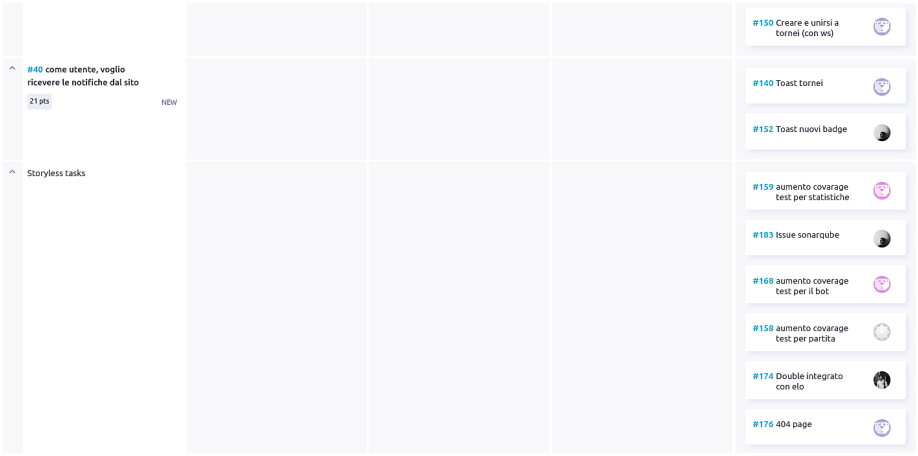

Figura 19: Backlog del quarto sprint - parte 2

3.5.3 Incremento

3.5.4 Definition of Done

3.5.5 Esempio di test fatti

3.5.6 Burndown

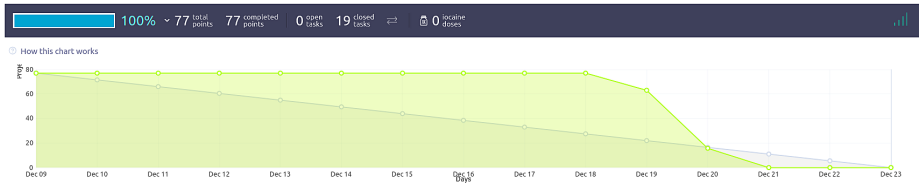

Figura 20: Burndown del quarto sprint

3.5.7 Retrospettiva

3.6 Release sprint

4 Descrizione del processo

Il team ha deciso di comune accordo e su richiesta del *Product Owner* di dedicare al progetto quattro sprint di due settimane ciascuno, in aggiunta allo sprint 0, dedicato alle attività di teambuilding e alla familiarizzazione con l'ambiente di sviluppo, e al release sprint, in cui il focus è andato sulla risoluzione dei bug, qualità del codice e stesura di questo report finale.

4.1 Il team

- **Product Owner**: Diego Barbieri
- **Scrum Master**: Lorenzo Peronese
- **DevOps**: Samuele Musiani
- **Developer frontend**: Emanuele Argonni
- **Developer backend**: Fabio Murer
- **Developer backend**: Omar Ayache

Il team è composto da sei membri ed è la prima volta che lavoriamo tutti insieme su un progetto di questa portata. Sebbene non ci conoscessimo tutti a vicenda inizialmente, alcuni di noi avevano già collaborato durante precedenti attività, sia universitarie che non. Durante gli sprint abbiamo avuto modo di approfondire la conoscenza reciproca in un contesto diverso dal solito, rafforzando il rapporto e favorendo una collaborazione più efficace.

Prima di iniziare lo sprint 0, il team si è riunito di persona per definire i ruoli; questi sono stati rispettati per la maggior parte, ma con un approccio flessibile: lo Scrum Master e il Product Owner hanno aiutato nello sviluppo del frontend, mentre il DevOps ha supportato sia il frontend che il backend quando necessario. Questa versatilità si è rivelata vincente e ha permesso di ottimizzare il lavoro e affrontare le sfide che si sono presentate in modo più efficace.

4.2 Teambuilding

Per lo sprint 0 il team si è riunito in due giornate diverse per partecipare alle due attività di teambuilding: *Scrumble* e *Escape The Boom!*.

Il primo gioco, *Scrumble*, si è rivelato inizialmente complicato da comprendere; tuttavia, una volta avviata l'attività, i membri del team hanno rapidamente preso il ritmo. Nonostante gli obiettivi prefissati del gioco non siano stati minimamente raggiunti, l'esperienza si è rivelata piacevole e coinvolgente. Il team ha avuto l'opportunità di familiarizzare con la metodologia Scrum, un "assaggio" pratico di come funziona il lavoro in sprint.

Il secondo gioco, *Escape The Boom!*, è stato più intuitivo da avviare ma non meno impegnativo. Il team si è suddiviso in due gruppi: il primo era incaricato di leggere e interpretare le istruzioni, mentre il secondo aveva il compito di disinnescare la bomba virtuale. Anche in questo caso, il gruppo ha trascorso un'ora divertente e stimolante, sebbene con risultati altrettanto modesti.

Di seguito la tabella dell'autovalutazione di *Scrumble*, compilata da tutti i partecipanti:

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

Figura 21: Tabella di autovalutazione del gioco *Scrumble*

4.3 Gitinspector

Le statistiche fornite dal Professor/Stakeholder Missiroli, generate tramite Gitinspector, non risultano essere accurate (il numero di commit segnalato da Gitinspector è di molto inferiore al numero reale). Non siamo riusciti a determinare con certezza se il problema risieda nel nostro repository o nel software stesso. Tuttavia, dopo aver effettuato ulteriori analisi indipendenti utilizzando Gitinspector, abbiamo riscontrato gli stessi risultati non corretti. Per completezza, i dati originali sono disponibili su `GitLab`.

In questo report, vengono confivise invece le statistiche del repository ottenute tramite `git-quick-stats`, un altro strumento di analisi.

Si segnala che i dati riportati non includono la stesura di questo documento.

author	Diego Barbieri PO	Lorenzo Peronese SM	Samuele Musiani DEV	Emanuele Argonni DEV	Fabio Murer DEV	Omar Ayache DEV	Total
lines_changed	2,547	4,923	9,885	14,854	18,060	5,652	55,921
%lines_changed	4.55%	8.80%	17.68%	26.56%	32.30%	10.11%	100%
commits	64	61	169	125	108	66	593
%commits	10.79%	10.29%	28.50%	21.08%	18.21%	11.13%	100%
insertions	4,513	7,483	16,278	18,808	20,876	8,417	76,375
%insertions	5.91%	9.80%	21.31%	24.63%	27.33%	11.02%	100%
deletions	1,966	2,560	6,393	3,954	2,816	2,765	20,454
%deletions	9.61%	12.52%	31.26%	19.33%	13.77%	13.52%	100.00%

Figura 22: Dati grezzi ricavati da git-quick-stats

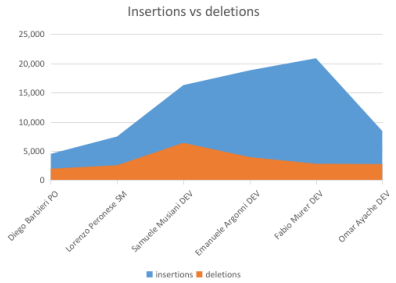

Figura 23: Rapporto tra linee inserite ed eliminate, le righe persistenti complessive sono oltre 55 mila

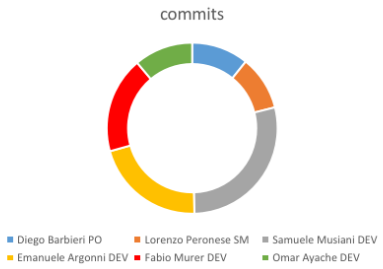

Figura 24: Numero di commit nel repository, per un totale di 593 e una media di circa 15 al giorno

4.4 Monitoraggio delle ore

Per il monitoraggio delle ore spese nel progetto non abbiamo utilizzato software di logging. Questa scelta è stata motivata dalla diversità di IDE utilizzati dai membri del team, che ha reso difficile individuare uno strumento compatibile per tutti. Inoltre, abbiamo provato inizialmente a utilizzare un timer per tracciare le ore, ma ci siamo resi conto che, non essendo un lavoro a tempo pieno, era comune dedicarsi ad altre attività durante le sessioni di scrittura del codice, questo rendeva il timer uno strumento poco adatto alla nostra situazione.

Abbiamo quindi optato per un approccio più semplice e flessibile, utilizzando un foglio Excel condiviso. Ogni membro del team ha segnato manualmente le ore effettivamente dedicate al progetto, garantendo una panoramica chiara e trasparente del tempo complessivamente investito. Di seguito quindi la tabella appena citata.

	Sprint 0	Sprint 1	Sprint 2	Sprint 3	Sprint 4	Release	Total
Diego Barbieri PO	9	16	13	29	34	19	120
Lorenzo Peronese SM	8	13	12	31	40	20	124
Samuele Musiani DEV	7	23	25	41	35	5	136
Fabio Murer DEV	7	22	25	36	40	5	135
Emanuele Argonni DEV	7	19	27	43	30	6	132
Omar Ayache DEV	6	18	22	38	35	7	126

Figura 25: Dati grezzi raccolti durante lo sviluppo

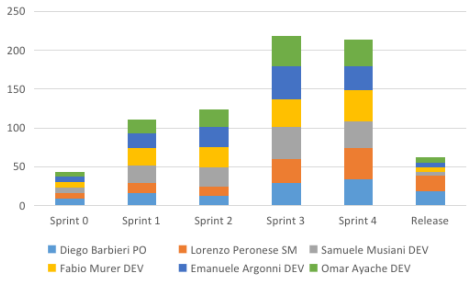

Figura 26: Ore di lavoro per ogni sprint, il totale è 773 ore (32 interi giorni!)

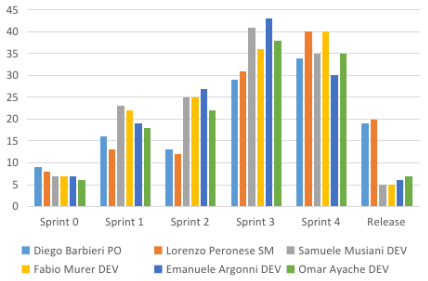

Figura 27: Più nel dettaglio, ore di lavoro per membro del team per sprint

4.5 Strumenti di comunicazione

Per comunicare tra noi abbiamo deciso di usare MatterMost, una piattaforma di chat e condivisione interamente open-source. Come per gli altri servizi, anche MatterMost è stato self-hostato sotto il dominio vezgammon.it per avere un maggiore controllo sui dati. Questa scelta ha permesso al team di gestire in autonomia l'infrastruttura di comunicazione, evitando dipendenze da servizi esterni e assicurando la personalizzazione dell'ambiente in base alle necessità del progetto.

MatterMost è stato utilizzato per la comunicazione quotidiana, la condivisione di aggiornamenti e l'organizzazione degli incontri di persona, che si sono svolti con frequenza. Gli incontri hanno avuto luogo presso il laboratorio del gruppo ADMstaff, situato nel seminterrato del Dipartimento di Informatica, in Mura Anteo Zamboni 7.

Il laboratorio è diventato il punto di riferimento per il team, che vi si è riunito interi pomeriggi per daily scrum, sessioni di pair programming e discussioni sull'avanzamento generale del progetto. Questa modalità di lavoro ha favorito una collaborazione continua e diretta, contribuendo significativamente alla coesione del gruppo e al progresso del progetto.

Come strumento di comunicazione secondario è stato usato Telegram per comunicazioni meno "ufficiali" e sopratutto per ricevere notifiche relative allo stato dei vari servizi tramite *Status* (vedi 2.1.7).

4.6 Utilizzo di Large Language Models

Abbiamo integrato strategicamente diversi modelli di intelligenza artificiale generativa nello sviluppo del progetto. Nello specifico, abbiamo impiegato GitHub Copilot per l'autocompletamento del codice ripetitivo direttamente nell'IDE, Claude per la risoluzione di problematiche tecniche specifiche e ChatGPT per la stesura della documentazione.

L'approccio all'utilizzo di questi strumenti è stato mirato: per ogni problema, abbiamo scelto di impiegare l'AI o come supporto iniziale nell'esplorazione delle possibili soluzioni o come strumento per risolvere un particolare problema o perfezionare un'implementazione già esistente, evitando la sovrapposizione dei due metodi.

Nonostante questi strumenti abbiano contribuito a incrementare la produttività del team, il loro impiego ha richiesto particolare attenzione: le "allucinazioni" e gli errori nel codice generato hanno reso necessaria un'attenta verifica delle risposte e in diverse occasioni il tempo dedicato al debugging del codice generato ha superato il risparmio di tempo ottenuto nella sua scrittura.

Con l'esperienza, siamo giunti a un equilibrio efficace che ci ha permesso di sfruttare le potenzialità di questi potenti strumenti mantenendo al contempo un controllo adeguato per cercare di arginarne i lati negativi.

4.6.1 Esempi di Prompt Utilizzati

Di seguito sono riportati alcuni esempi a titolo esemplificativo dei prompt utilizzati, organizzati per categoria:

- Fix this vue component error: `{...}`
- How do I refactor the code to make it work with tailwind themes?

- Given these SQL tables: `{...}` Make a SQL query to return the table `game` with usernames instead of `p1_id`, `p2_id`.
- PostgreSQL returns error: `syntax error at or near "FROM"` for the following query: `{...}`

- Reformulate this user story: `{...}` Make it detailed enough for developers but not too technical for the client.
- This user story is too challenging: `{...}` Break it down into 3/4 smaller and more manageable user stories.

- How can I make this retrospective more objective and data-based? `{...}`
- How can I make this documentation more concise while maintaining all essential information? `{...}`

4.7 SonarQube

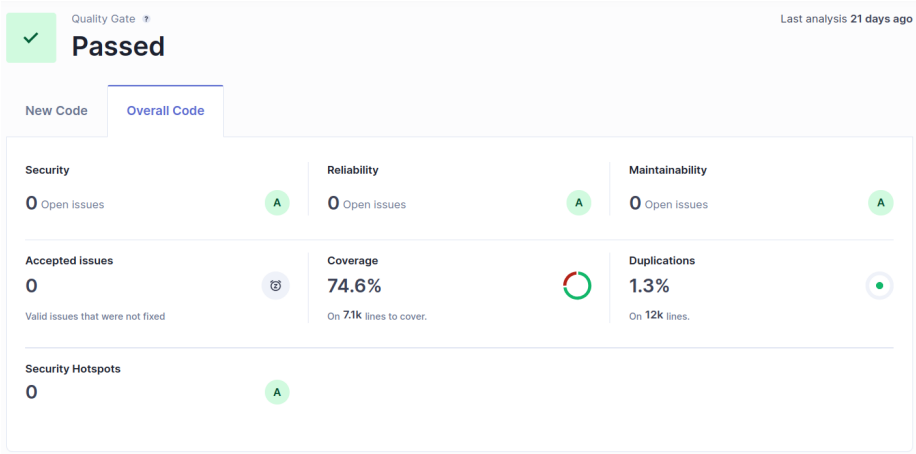

Figura 28: Vista generale di SonarQube, che mostra un riepilogo delle metriche del progetto, tra cui copertura del codice, vulnerabilità, codice duplicato e molto altro.
L'obiettivo di una copertura del 50% è stato abbondantemente superato; tuttavia le aree di codice meno coperte dai test riguardano principalmente le API e le parti che fanno uso dei web socket, in quanto più difficili da testare.

Figura 29: Grafico delle issue rilevate nel tempo: il picco registrato intorno al 15 novembre è legato all'inclusione di due file HTML di GitInspector simili, che ha portato SonarQube a rilevare numerosi problemi di duplicazione. Dopo aver individuato la causa, abbiamo escluso quei file dall'analisi, riportando i valori alla normalità. Durante il resto dello sviluppo, il numero delle issue è stato mantenuto sotto controllo, risolvendo i problemi a intervalli regolari.

5 Demo 3 min

Link qui