VezGammon - Report finale Team 1

Progetto di Ingegneria del Software

Lorenzo Peronese - Scrum Master

23 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

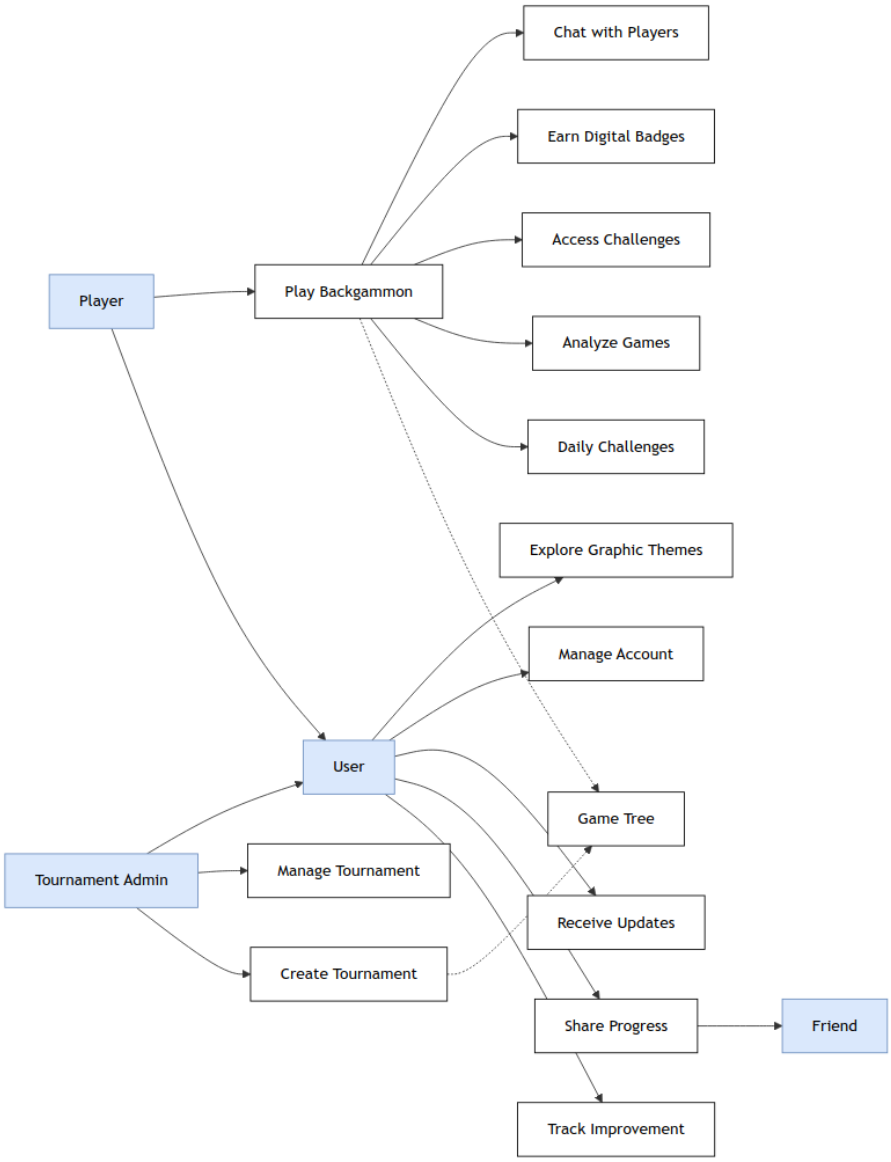

Figura 2: Diagramma UML use-case

1.3.2 Deployment

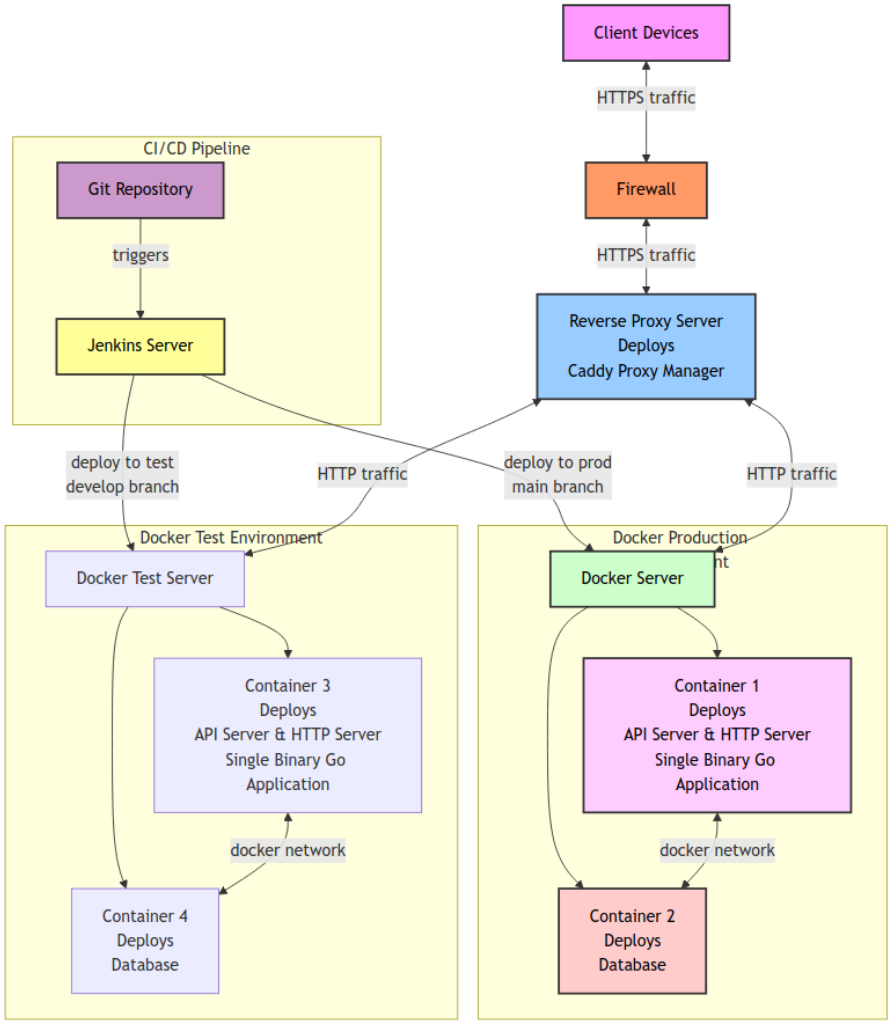

Figura 3: Diagramma UML di deployment

1.3.3 Classi

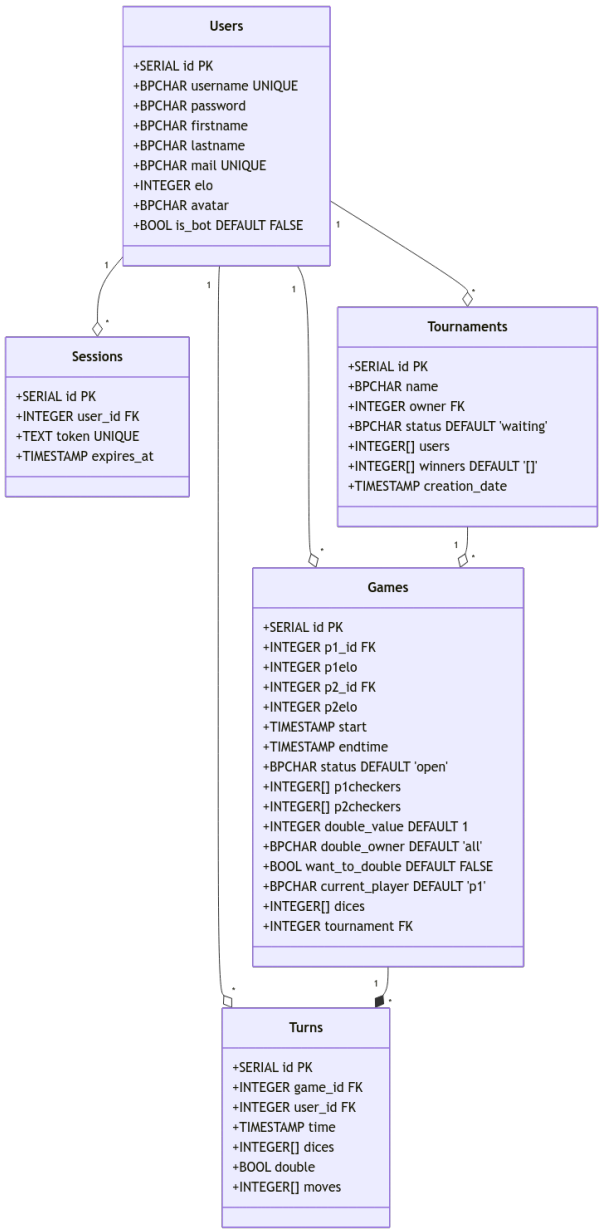

Figura 4: Diagramma UML delle classi

1.3.4 Tornei

Figura 5: Diagramma UML dei tornei

2 Infrastruttura e tecnologia utilizzata

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 personalizzazione e per 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 3.5, 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`

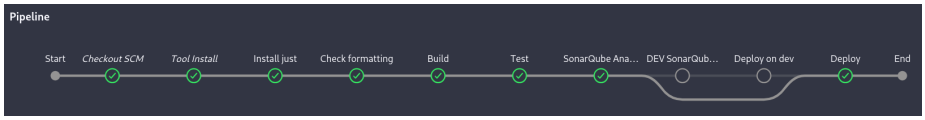

Figura 6: Pipeline di Jenkins: viene lanciata a ogni evento push su GitLab e controlla formattazione, build e test; sui commit delle branch *main* e *develop* viene anche eseguito il controllo statico del codice con SonarQube e il deploy sul relativo server, di produzione o di sviluppo.

2.1.5 SonarQube

Come descritto nella Sezione 3.7, 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

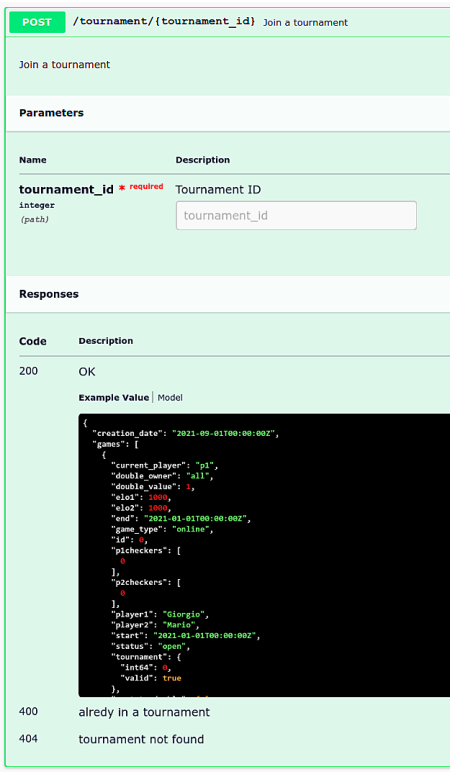

Figura 8: Nel dettaglio, l'interfaccia di Swagger per ogni API

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`

2.1.8 bgweb-api

Il team ha utilizzato `bgweb-api` come software open source di validazione delle mosse, un wrapper del software utilizzato da `GNU-Backgammon` compatibile con il linguaggio Go.

2.2 Stack Web

Frontend

- **Framework**: `Vue.js`
- **Libreria di styling**: `Tailwind CSS`

Backend

- **Linguaggio**: `Go`
- **Framework**: `Gin`

Database

- **Tipo**: `PostgreSQL`
- **Interfaccia web**: `Pgweb`

Comunicazione

- **Architettura**: `REST API`
- **Documentazione**: `Swagger` (vedi 2.1.6)

2.3 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 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.

3.1 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 ha permesso di ottimizzare il lavoro e affrontare le sfide che si sono presentate in modo più efficace.

3.2 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 9: Tabella di autovalutazione del gioco *Scrumble*

3.3 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 10: Dati grezzi ricavati da git-quick-stats

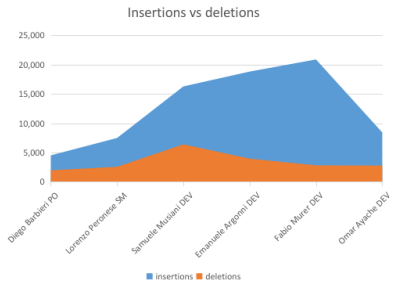

Figura 11: Rapporto tra linee inserite ed eliminate, le righe persistenti complessive sono oltre 55 mila

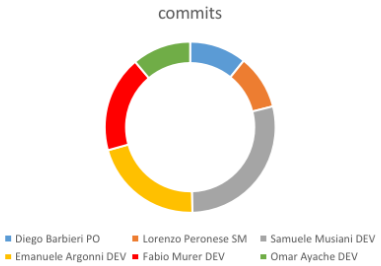

Figura 12: Numero di commit nel repository, per un totale di 593 e una media di circa 15 al giorno

3.4 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 13: Dati grezzi raccolti durante lo sviluppo

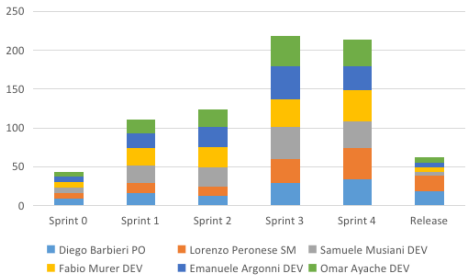

Figura 14: Ore di lavoro per ogni sprint, il totale è 773 ore (32 interi giorni!)

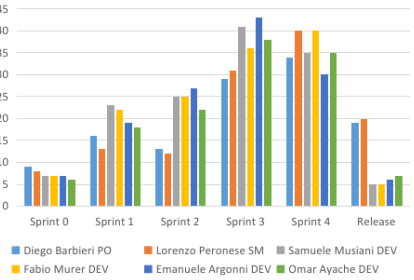

Figura 15: Più nel dettaglio, ore di lavoro per membro del team per sprint

3.5 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).

3.6 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.

3.6.1 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? `{...}`

3.7 SonarQube

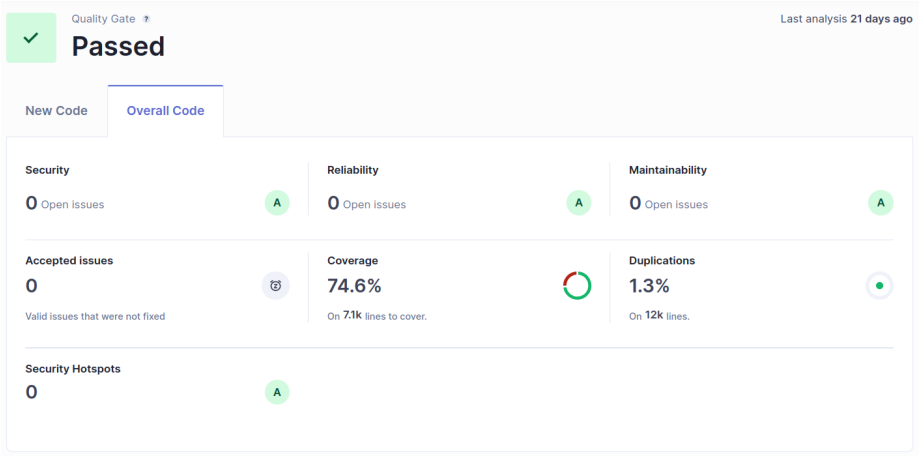

Figura 16: Vista generale di SonarQube, che mostra un riepilogo delle metriche del progetto, tra cui copertura del codice, vulnerabilità, codice duplicato e molto altro.
L'obiettivo di una copertura del 50% è stato abbondantemente superato; tuttavia le aree di codice meno coperte dai test riguardano principalmente le API e le parti che fanno uso dei web socket, in quanto più difficili da testare.

Figura 17: 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.

4 Processo Scrum

4.1 Definition of Done

Durante lo sprint 0 il *Product Owner* ha definito i requisiti per dichiarare una user story come conclusa; questi requisiti sono stati accettati dal team e sono rimasti invariati durante tutta la durata dello sviluppo.

Vengono quindi riportati di seguito:

4.1.1 Requisiti della Code Base

- Qualità del Codice
  - Tutto il codice deve seguire le convenzioni di denominazione in camelCase.
  - Ogni modulo deve avere e superare i propri test unitari (e mock, se sviluppato con TDD).
  - Nessun codice critico o errori gravi rilevati dagli strumenti di analisi statica (es. SonarQube).
  - Per le principali pagine del front-end, la funzionalità di base deve essere testata tramite test front-end.
- Documentazione
  - Tutte le API devono fornire un'anteprima con Swagger e includere test specifici per le API.
  - README principale aggiornato con le funzionalità più recenti.
- Merge Request
  - Ogni merge request (MR) deve essere approvata dallo Scrum Master (SM) o dal Product Owner (PO).
  - La MR deve superare tutti i test nella pipeline di testing di Jenkins.
- Interfaccia Utente
  - Soddisfa gli standard WCAG 2.1 AA: compatibile con tutti i browser moderni.
  - Responsiva su laptop, tablet e smartphone (solo visualizzazione orizzontale).
- Prestazioni
  - L'app mantiene prestazioni accettabili (tempo di caricamento inferiore a 2 secondi per l'interfaccia principale del gioco).

4.1.2 Requisiti di Deployment

- Compatibilità
  - L'applicazione funziona correttamente negli ambienti di sviluppo e produzione.
  - L'applicazione deve richiedere un file di configurazione, facile da configurare e compatibile con l'architettura di deployment.
- Build e Rilascio
  - L'applicazione può essere compilata senza errori o avvisi.
  - L'app è configurata per l'integrazione e il deployment continui, con test automatizzati per ogni merge.
- Monitoraggio e Logging
  - Sono attivi il monitoraggio degli errori in tempo reale e il logging front-end e back-end
  - I log catturano eventi critici come l'inizio delle partite, le disconnessioni e gli errori dell'utente.

4.2 Definition of Ready

Anche in questo caso, il documento *Definition of Ready* è stato redatto dal *Product Owner* all'inizio dello sviluppo, accettato dal resto del team e durante gli sprint è rimasto pressocchè invariato.

Di seguito i requisiti che ogni user story deve avere per essere considerata *ready*:

- Il PBI è scritto in modo chiaro e conciso.
- Sono stati definiti criteri di accettazione testabili.
- Sono state identificate e risolte tutte le dipendenze.
- Tutti gli stakeholder hanno approvato il PBI.
- Le caratteristiche relative alla UI sono state discusse con il PO.
- Sono presenti requisiti tecnici (es. database, API).
- La documentazione preliminare (design, flussi di gioco) è completa.
- È stata fatta una stima delle ore di lavoro.
- Sono stati definiti i requisiti di accessibilità e usabilità.
- È stato convalidato con un piccolo gruppo di utenti (se necessario).

4.3 Sprint 0

Oltre alle attività di teambuilding (vedi 3.2), i membri del gruppo hanno iniziato in questa fase a studiare il funzionamento e la documentazione dei software scelti per lo sviluppo.

4.4 Sprint 1

4.4.1 Goal

Come si può vedere dallo snapshot di Taiga, lo sprint goal era piuttosto ambizioso e prevedeva 3 principali obiettivi:

- creazione e gestione degli account;
- accesso a una partita;
- visualizzazione dei progressi di gioco.

4.4.2 Backlog

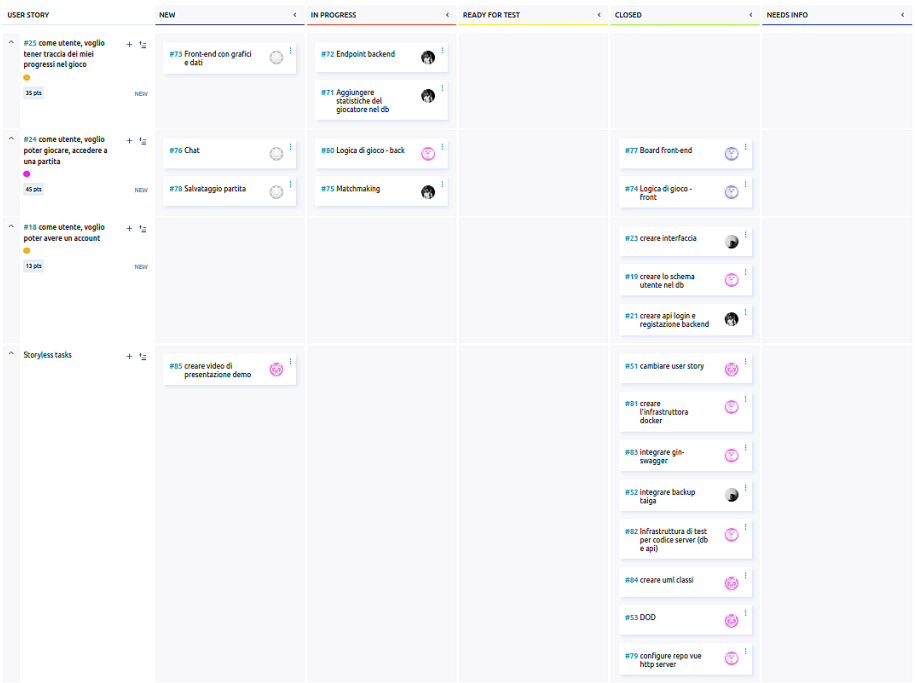

Figura 18: Backlog del primo sprint

4.4.3 Incremento

Alla fine dello sprint, il team ha completato con successo la user story che riguarda creazione e gestione dell'account (con autenticazione) ed era inoltre possibile giocare una partita locale molto semplice, tuttavia l'implementazione delle altre funzionalità chiave ha incontrato significative difficoltà:

- sottostima del carico di lavoro;
- inefficace assegnazione delle user stories, troppo impegnative e generiche;
- complessità di integrazione con le nuove tecnologie;
- rallentamenti causati dal self-hostating dell'intero ambiente.

Un fattore critico è stato l'utilizzo di `Bgweb-api` per permettere di giocare contro un agente intelligente, che ha introdotto vincoli architetturali nella rappresentazione delle classi, in particolare le proprietà della partita. Questa scelta ha generato dipendenze che hanno rallentato lo sviluppo del backend anche durante gli sprint successivi.

4.4.4 Esempio di test fatti

Creazione del profilo

- Obiettivo: testare la registrazione di un utente
- Scenario: un utente vuole creare un profilo utente registrandosi al sito
- Passaggi:
  - navigare alla home del sito
  - Premere *Sign up*
  - Inserire tutti i campi correttamente
  - Premere *Register*
  - effettuare il login
- Risultato atteso: l'utente deve essere creato correttamente e il login deve andare a buon fine

4.4.5 Burndown

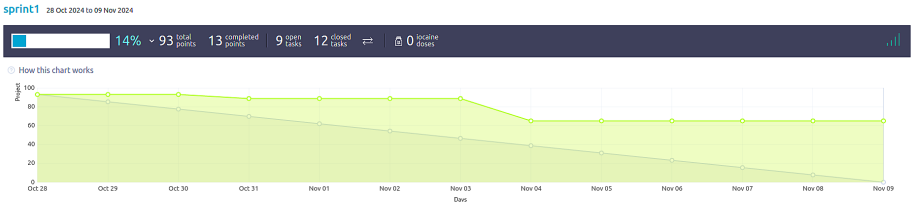

Figura 19: Burndown del primo sprint

4.4.6 Retrospettiva

Cards	Diego PO	Lorenzo SM	Samuele D	Fabio D	Emanuele D	Omar D
Scrum Master	😡	😡	😡	😡	😐	😐
Product Owner	😡	😐	😐	😐	😐	😐
Development Team	😐	😡	😡	😐	😐	😐
Product Backlog	😃	😃	😐	😃	😃	😃
Sprint Planning	😡	😡	😐	😡	😡	😡
Sprint Goal	😡	😡	😐	😐	😐	😐
Self Menagement	😃	😐	😃	😃	😃	😐

Figura 20: Voti della retrospettiva del primo sprint

Commenti personali

- Product Owner - Diego Barbieri:
  Sotto-stima del carico di lavoro, specialmente riguardante la tecnologia per

il back-end, che porterà ad un elevato debito tecnico del secondo sprint. Poco dialogo e male suddivisione dei compiti fra i developers. User story ben definite, task troppo vaghe. Suddivisione impari del carico di lavoro: alcuni membri hanno lavorato più di altri. Veramente ottimo la gestione dell'ambiente di lavoro: server di deploy funzionante con un ottimo CI/CD.

- Scrum Master - Lorenzo Peronese:
  Ottima gestione dell'ambiente di lavoro self-hostato, ma user stories troppo complesse e impegnative, soprattutto relative al back-end. Cattiva organizzazione della suddivisione delle task, anche a causa mia che ho dato scadenze poco precise e ho seguito troppo sommariamente lo sviluppo, senza aver capito a pieno lo scopo del mio ruolo.

- Developer - Samuele Musiani:
  Sotto-stima del carico di lavoro, in particolare troppo per il backend e relativamente poco per il front-end. Il team per il front si è trovato bloccato da mancanza di features nel back.

- Developer - Fabio Murer:
  Sotto-stima del carico di lavoro, sprint molto impegnativo a causa di user story troppo generali richiedenti molto lavoro. Continui problemi nel database mi hanno portato ad accumulare un ritardo nello sviluppo di features nella parte back-end.

- Developer - Emanuele Argonni:
  Carico di lavoro sottostimato, abbiamo un elevato debito per il prossimo sprint. Era da pensare meglio la suddivisione delle user story considerati i 3 sprint. PO e SM dovrebbero gestire meglio il team durante l'intero periodo dello sprint evitando di assegnare attività gli ultimi giorni prima della fine dello sprint.

- Developer - Omar Ayache:
  Oltre ai commenti e riflessioni dei colleghi che condivido, vorrei che al prossimo sprint si prendesse in considerazione una specie di misura (i.e. i giorni di lavoro o le ore) una misura quantitativa che associ task/tempo di lavoro. Questo dovrebbe comprendere anche la "tassa" della nuova tecnologia utilizzata, per esempio il capire come funziona la gestione del db con pSQL (oppure i possibili problemi che possono nascere). I vantaggi dovrebbero essere che in questo modo riusciamo a valutare meglio sia il carico di lavoro ma anche il peso (punti) da assegnare ad ogni US.

Prolematiche comuni e proposte per risolverle

Durante questo sprint, il carico di lavoro front-end è stato significativamente inferiore rispetto a quello back-end, causando tempi di inattività per gli sviluppatori front-end in attesa del completamento delle componenti server. Per ottimizzare gli sprint futuri, sarà fondamentale:

- bilanciare meglio la distribuzione delle attività tra front-end e back-end;

- garantire un flusso di sviluppo continuo e efficiente.

Il sovraccarico di user stories assegnate a questo sprint ha generato un debito tecnico considerevole che dovrà essere gestito nello sprint successivo. Di conseguenza, le attività inizialmente pianificate dovranno essere riprogrammate. Product Owner e Scrum Master si incontreranno lunedì per valutare strategie che evitino l'aggiunta di uno sprint supplementare rispetto alla pianificazione iniziale. Qualora l'estensione risultasse inevitabile, si procederà con una nuova distribuzione delle user stories.

4.5 Sprint 2

4.5.1 Goal

Lo sprint goal prevedeva lo sviluppo di *core features* del gioco:

- implementazione di un'interfaccia grafica strutturata;
- sviluppo di modalità di gioco locali (giocatore vs giocatore e giocatore vs bot) e online;
- personalizzazione dei temi grafici;
- implementazione del sistema di tracciamento dei progressi di gioco.

4.5.2 Backlog

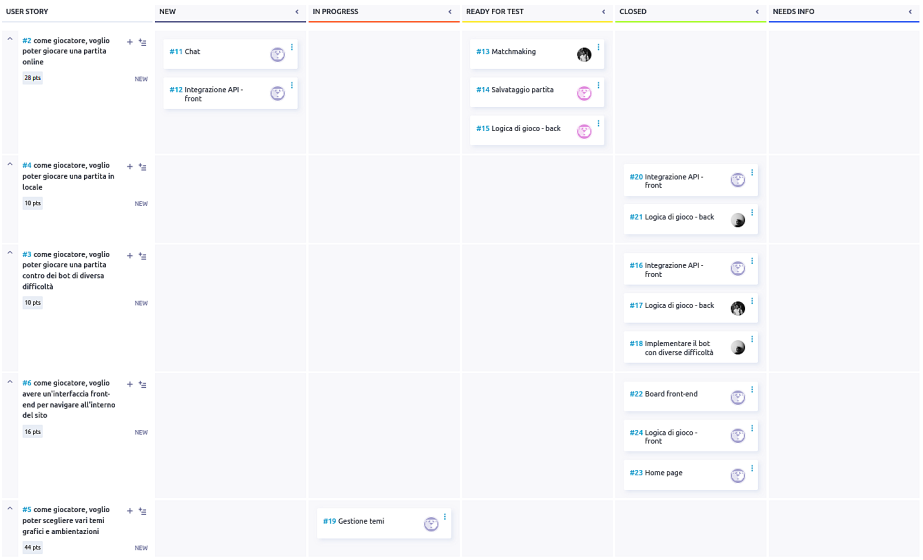

Figura 21: Backlog del secondo sprint - parte 1

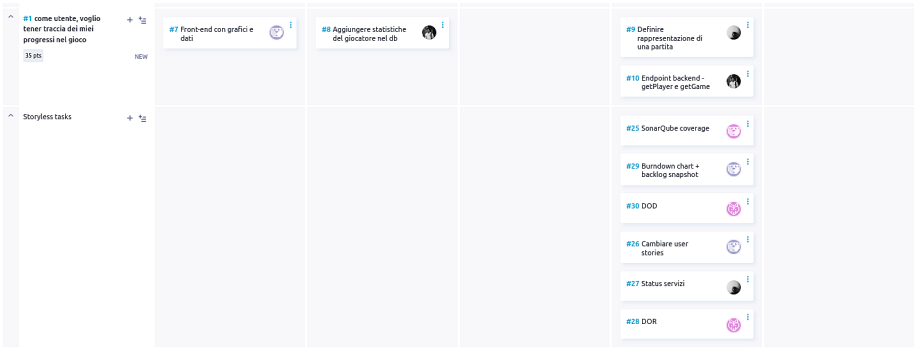

Figura 22: Backlog del secondo sprint - parte 2

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.

4.5.3 Incremento

Questi sono i risultati ottenuti dal team, le user stories completate:

- interfaccia front-end strutturata per la navigazione del sito;
- partita locale (su un unico schermo);
- partita contro bot di diversa difficoltà.

Le storie non completate, quindi il debito tecnico, sono state la gestione della partita online (sempre a causa del validatore di mosse e della difficoltà ad interfacciarsi ad esso), l'implementazione di altri temi grafici e, ancora una volta, la dashboard con i progressi di gioco.

4.5.4 Esempio di test fatti

Giocare una partita in locale

- Obiettivo: Verificare che si possa creare e giocare una partita in locale
- Scenario: Un utente crea una partita in locale, si effettuano due turni e si chiude la partita abbandonando
- Passaggi:
  - un utente crea una partita in locale
  - effettua il turno 1
  - effettua il turno 2

- chiude la partita abbandonando

- Risultato atteso: la partita viene creata con lo stesso giocatore come giocatore 1 e 2, i due turni si possono giocare dallo stesso giocatore, dopo l'abbandono la partita viene chiusa e lo stato viene messo a winp2

4.5.5 Burndown

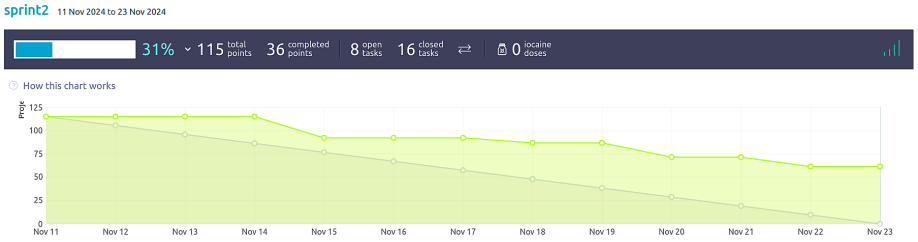

Figura 23: Burndown del secondo sprint

4.5.6 Retrospettiva

Cards	Diego PO	Lorenzo SM	Samuele D	Fabio D	Emanuele D	Omar D
Scrum Master	😃	😃	😃	😃	😃	😃
Product Owner	😃	😃	😃	😐	😃	😃
Development Team	😐	😃	😐	😐	😃	😐
Product Backlog	😃	😃	😃	😐	😐	😐
Sprint Planning	😐	😡	😐	😃	😐	😐
Sprint Goal	😐	😐	😐	😐	😐	😐
Self Menagement	😃	😃	😃	😃	😃	😐

Figura 24: Voti della retrospettiva del secondo sprint

Commenti personali

- Product Owner - Diego Barbieri:
  Nel complesso, penso che il team abbia lavorato al meglio delle proprie capacità. Sono stati parzialmente risolti i problemi di comunicazione e di suddivisione del carico di lavoro. Penso che sia necessaria maggiore attenzione agli standard da adottare per le API tra front e back-end. Questo puo' essere risolto con una maggiore attenzione agli schemi di progettazione e alla documentazione. Sono stato negativamente colpito dal fatto che durante l'ultima sprint review, alcuni membri del team non avessero ben chiaro il significato di DOD e della sua importanza. Lo stesso e' successo quando un developer ha mergiato direttamente sul main parte dello sviluppo ancora non pienamente completato.
- Scrum Master - Lorenzo Peronese:
  Il team ha lavorato di più e meglio, con maggior comunicazione e aiuto

reciproco. I daily scrum sono stati strutturati meglio e sono passati da una frequenza irregolare a una cadenza fissa ogni due giorni; sono stati un momento utile per tutti di confronto e discussione. Unica nota negativa è stata ancora una volta una cattiva pianificazione dello sprint: 3 user stories su 6 complessive non sono state completate a causa di una stima sbagliata a inizio sprint e molti problemi imprevisti emersi durante queste due settimane; è stato richiesto molto impegno ai developers ma sono soddisfatto del modo in cui queste difficoltà sono state affrontate e superate.

- Developer - Samuele Musiani:
  Problemi nel back-end, soprattutto riguardo le api per backgammon. Sottostima dei tempi per completare certe US. Buona la scelta di spostare un developer del frontend nel backend.

- Developer - Emanuele Argonni:
  Purtroppo anche per questo sprint non siamo riusciti a portare a termine tutte la task, dato un incoveniente con le API del backend che non ritornavano tutte le mosse possibili Nella seconda settimana dello sprint grazie al pair-programming tra backend e frontend siamo riusciti a risolvere quasi tutti i problemi riscontrati, nonostante il tempo perso per il debugging. Buon lavoro dello scrum master che ha cercato di motivare i developer ed ha organizzato i daily scrum per permettere ai membri del team di confrontarsi

- Developer - Omar Ayache:
  Sottostima del lavoro e problemi riguardo gestione del bot per le mosse, ottimo l'utilizzo della strategia pair-programming per i prossimi sprint. Miglior presa di coscienza generale del carico di lavoro.

Prolematiche comuni e proposte per risolverle Dall'analisi delle opinioni raccolte emerge un miglioramento generale delle prestazioni del team rispetto allo sprint precedente, sebbene persistano alcune criticità nella stima del carico di lavoro e nella comunicazione interna. Alla luce di queste considerazioni, proponiamo le seguenti azioni per il prossimo sprint:

- effettuare piu' pair-programming per risolvere problemi di comunicazione. Questo permetterebbe di mettere in diretta relazione persone di diversi ruoli e di far capire meglio il lavoro che si sta svolgendo;

- suddividere il carico di lavoro includendo anche PO e SM in attività di sviluppo, per massimizzare e parallelizzare il lavoro. Questo non sottraendo tempo alle attività già previste dai loro ruoli.

4.6 Sprint 3

4.6.1 Goal

L'obiettivo di questo terzo sprint era dare un'importante accellerata allo sviluppo, pertanto il carico di lavoro è stato molto maggiore ma la situazione lo richiedeva. Per agevolare la buona riuscita dello sprint sono stati adottati diversi principi Agili:

- *Scrum Master* e *Product Owner* hanno dato una mano nella scrittura del codice, oltre a mantenere i loro ruoli abituali;
- i *Daily Scrum* sono stati più frequenti e articolati per pianificare e gestire al meglio il tempo a disposizione;
- *pair programming* tra sviluppatori front-end e back-end.

Le user stories previste per questo sprint riguardavano:

- dashboard con progressi di gioco;
- gestione di una partita online;
- replay e analisi delle partite giocate;
- gestione di nuovi temi;
- gestione di badge digitali al raggiungimento di determinati obiettivi;
- chat durante le partite;
- risorse di addestramento (tutorial e regole di gioco);
- condivisione dei progressi di gioco sui social.

4.6.2 Backlog

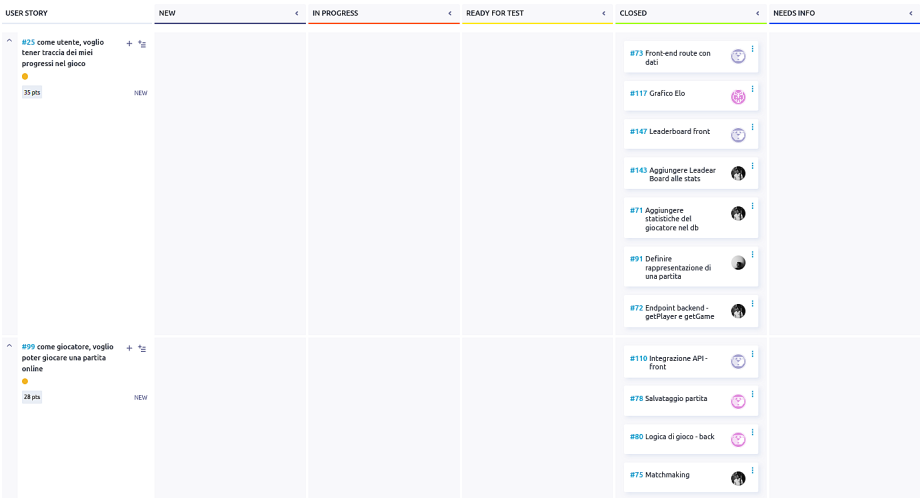

Figura 25: Backlog del terzo sprint - parte 1

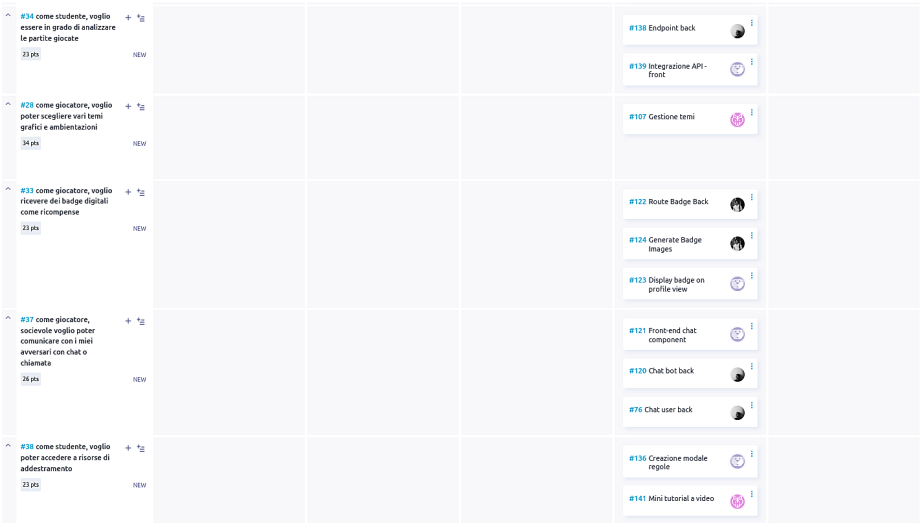

Figura 26: Backlog del terzo sprint - parte 2

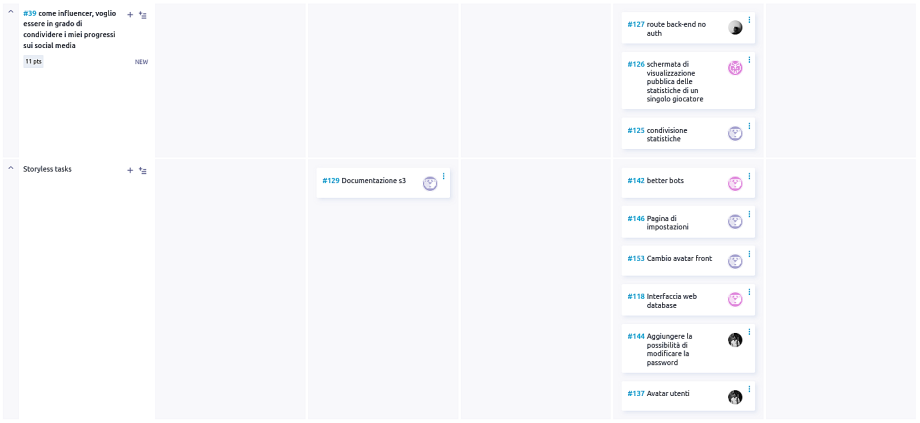

Figura 27: Backlog del terzo sprint - parte 3

4.6.3 Incremento

Questo sprint ha richiesto un enorme sforzo da parte di tutti, è stato di gran lunga quello con più user stories ma ciò nonostante è stato il primo in cui tutte le user stories sono state completate con successo, senza debito tecnico.

4.6.4 Esempio di test fatti

Generazione badge

- Obiettivo: testare la creazione di generazione del badge prima vittoria
- Scenario: un giocatore vince la prima partita
- Passaggi:
  - creare una partita
  - vincere la partita
- Risultato atteso: il giocatore deve avere il badge *first victory*

4.6.5 Burndown

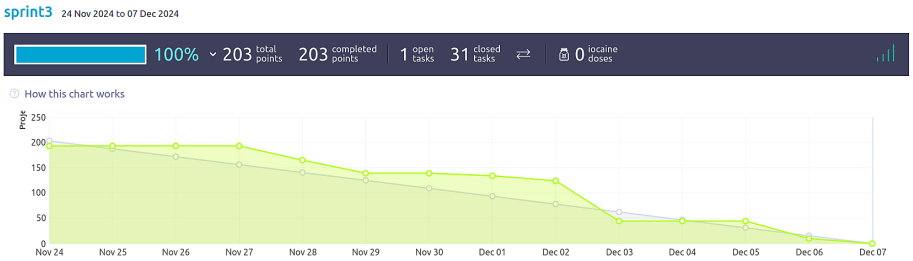

Figura 28: Burndown del terzo sprint

4.6.6 Retrospettiva

Cards	Diego PO	Lorenzo SM	Samuele D	Fabio D	Emanuele D	Omar D
Scrum Master	😃	😃	😃	😃	😃	😃
Product Owner	😃	😃	😃	😃	😃	😃
Development Team	😃	😃	😃	😃	😃	😃
Product Backlog	😃	😃	😃	😃	😃	😃
Sprint Planning	😃	😃	😃	😃	😃	😃
Sprint Goal	😃	😃	😃	😃	😃	😃
Self Menagement	😃	😃	😃	😃	😐	😃

Figura 29: Voti della retrospettiva del terzo sprint

Commenti personali

- Product Owner - Diego Barbieri:
  Ottimo sprint, in assoluto il migliore. Nonostante un primo momento di insicurezza generale sul superamento delle task, ci siamo trovato a meta' settimna con tutte le us quasi completate: e' risultato necessario un "daily scrum on steroids" per decidere ulteriori us da terminare. Alla fine sono molto soddisfatto del risultato ottenuto, il tutto grazie a un codice ben organizzato in componenti.

- Scrum Master - Lorenzo Peronese:
  Sprint eccellente, con il completamento di tutte le user stories assegnate. Gli imprevisti sono stati pochi e ben gestiti, grazie a uno sviluppo costante supportato dai tre daily scrum settimanali e da una comunicazione efficace, sia in chat che di persona in laboratorio. Il pair programming si è rivelato un grande successo, consentendo di risolvere molte problematiche in modo rapido ed efficace.

- Developer - Samuele Musiani:
  Sprint decisamente positivo sotto molteplici aspetti. Il team di sviluppo ha dimostrato un'ottima cooperazione che ha portato al completamento di quasi tutte le US con un anticipo notevole rispetto alle tempistiche preventivate. Il team ha inoltre dimostrato una completa padronanza degli strumenti utilizzati, potendo quindi focalizzare le proprie risorse principalmente sulle attività di implementazione, ottimizzando così l'efficienza dell'intero processo di sviluppo. Questo sprint testimonia la maturità del team e pone solide basi per il proseguimento del progetto.

- Developer - Emanuele Argonni:
  Sicuramente il migliore sprint fino ad ora, siamo riusciti a completare quasi tutte le task assegnate, nonostante avessimo un importante debito derivato dagli sprint precedenti. Il team dei developer ha lavorato bene e si è dimostrato molto collaborativo. Il lavoro di squadra è stato fondamentale per il raggiungimento degli obiettivi. Il pair programming è stato molto utile per l'integrazione backend-frontend e per risolvere i problemi che si sono presentati durante lo sprint. Terminate le task assegnate, ci siamo dedicati al refactor del codice per migliorare la chiarezza e la manutenibilità del codice.

- Developer - Omar Ayache:
  Sprint eccezionale, con obiettivi quasi completamente raggiunti. Diversi meeting scrum dinamici hanno permesso di ridefinire ulteriori attività. Il team ha dimostrato una significativa crescita, padroneggiando con competenza gli strumenti e comprendendo la loro complessità di utilizzo. L'organizzazione del codice in componenti ben strutturate ha contribuito al successo complessivo.

4.7 Sprint 4

4.7.1 Goal

Il goal di questo ultimo sprint è stato dedicato all'ultima *core feature*, ovvero i tornei, sia con altri giocatori che con bot di diversa difficoltà. Inoltre, in queste due settimane l'obiettivo era quello di rendere l'intera app responsive e usufruibile da cellulare e ricevere notifiche in app, ad esempio alla ricezione di nuovi badge o durante la partecipazione a un torneo.

4.7.2 Backlog

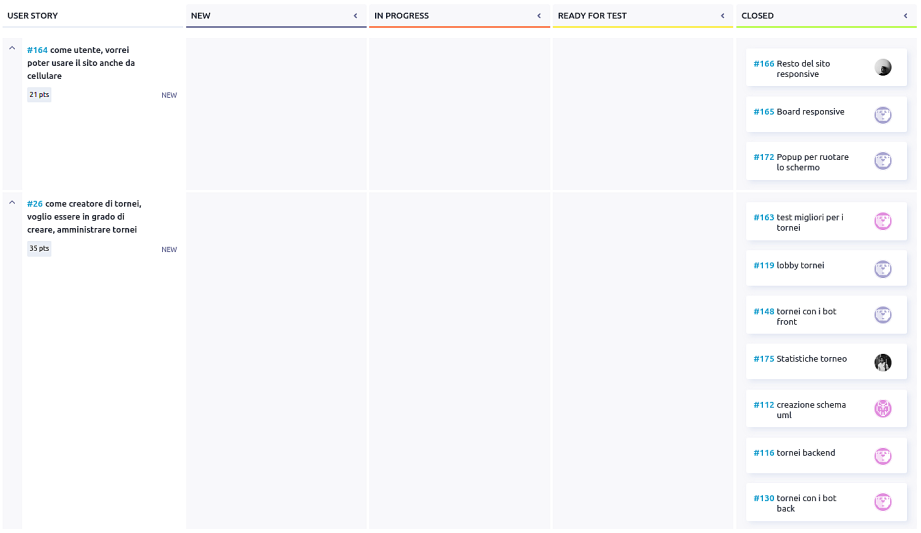

Figura 30: Backlog del quarto sprint - parte 1

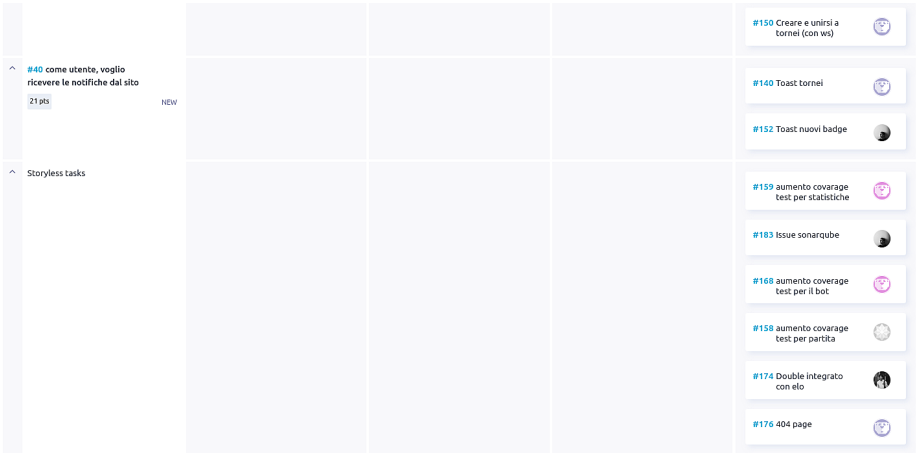

Figura 31: Backlog del quarto sprint - parte 2

4.7.3 Incremento

Anche in questo sprint tutte le task sono state completate con successo, non scontato data la sovrapposizione con l'inizio della sessione invernale d'esami.

4.7.4 Esempio di test fatti

Verificare che il torneo venga creato correttamente

- Obiettivo: Verificare un utente possa creare un torneo e questo sia creato correttamente
- Scenario: Un utente crea un torneo
- Passaggi:
  - navigare all'interfaccia di creazione tornei
  - inserire il nome del torneo che si vuole creare
  - creare un torneo utilizzando l'apposito bottone
- Risultato atteso: un nuovo torneo deve essere creato con i seguenti campi: l'utente che lo ha creato come owner, come unico utente l'owner e come stato waiting

4.7.5 Burndown

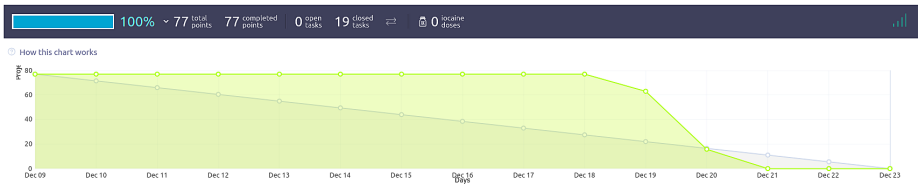

Figura 32: Burndown del quarto sprint

4.7.6 Retrospettiva

Cards	Diego PO	Lorenzo SM	Samuele D	Fabio D	Emanuele D	Omar D
Scrum Master	😃	😃	😃	😃	😃	😃
Product Owner	😃	😃	😃	😃	😃	😃
Development Team	😃	😃	😃	😃	😃	😃
Product Backlog	😃	😃	😃	😃	😃	😃
Sprint Planning	😃	😃	😃	😃	😃	😃
Sprint Goal	😃	😃	😃	😃	😃	😃
Self Menagement	😃	😃	😃	😃	😃	😃

Figura 33: Voti della retrospettiva del quarto sprint

Commenti personali

- Product Owner - Diego Barbieri:
  Sprint molto positivo, quasi quanto il precedente, ovviamente con un cari-

co di lavoro minore. Simao riusciti a portare tutto al termine, grazie ad un ottima organizzazione del team, maturata durante tutto questo viaggio.

- Scrum Master - Lorenzo Peronese:
  Durante lo sprint il team ha lavorato bene, portando a termine tutte le user stories nei tempi previsti nonostante l'inizio degli esami. Il pair programming tra frontend e backend e i daily scrums sono sono diventate abitudini Agili e si sono rivelate fondamentali per gestire il doppio impegno studio-lavoro senza sacrificare la qualità del codice. Questo e il precedente sprint dimostrano il miglioramento che c'è stato rispetto alle prime settimane, il team è riuscito a prendere il ritmo giusto e concludere il lavoro nel migliore dei modi.

- Developer - Samuele Musiani:
  Sprint positivo, soprattutto a seguito della conclusione di tutte le US assegnate. Il team ha lavorato bene nonostante l'inzio degli esami universitari, portando a termine ogni compito nel tempo previsto. Si riconfermata la piena padronanza degli strumenti utilizzati e la capacità di coordinarsi per portare a termine tutte le task più complesse che richiedevano una collaboraione tra frontend e backend.

- Developer - Fabio Murer:
  Sprint positivo, grazie ai daily scrum frequenti e il lavoro a stretto contatto tra backend e frontend questo sprint è risultato molto positivo permettendoci di risolvere velocemente le task e di risolvere un gran numero di bug.

- Developer - Emanuele Argonni:
  Questo sprint, come il precedente, ha evidenziato un netto miglioramento rispetto alle prime settimane del progetto. Siamo riusciti a portare a termine tutte le US assegnate nei tempi stabiliti. Il pair programming tra backend e frontend ha garantito la collaborazione di tutti i developer e ha facilitato la risoluzione di bug. L'ottima sinergia tra i membri del team e la comunicazione costante ci hanno permesso di affrontare le difficoltà e di gestire al meglio il carico di lavoro.

- Developer - Omar Ayache:
  Questo sprint ha evidenziato l'ottima coesione del team e il rispetto dei principi Agile, con comunicazione efficace e collaborazione continua. Il pair programming e i daily scrum ci hanno permesso di affrontare le sfide e completare tutte le user stories nei tempi previsti, mantenendo alta la qualità del lavoro anche durante il periodo intenso degli impegni personali e lavorativi.

4.8 Release sprint

Il release sprint è stato dedicato al test intensivo dell'intero sito e alla risoluzione dei bug, con l'obiettivo di consegnare ai clienti un prodotto solido e interamente funzionante.

Inoltre questo periodo è stato dedicato alla stesura del presente documento, riassuntivo dell'intero progetto.

5 Video demo

A questo link si trova la demo di VezGammon™ girata dal *Product Owner*, in circa 3 minuti vengono riassunte le funzionalità di base e ci si può fare un'idea del funzionamento.

Il prodotto è online e raggiungibile all'url `https://vezgammon.it`.

6 Conclusione???