Rapporto del TEAM 12

Tweet Analysis

Cheikh Ibrahim · Zaid
PO Operativo
Matricola: 0000974909

Xia · Tian Cheng
Scrum master
Matricola: 0000975129

Lee · Qun Hao Henry
Developer
Matricola: 0000990259

Paris · Manuel
Developer
Matricola: 0000997526

Anno accademico
2022 – 2023

Corso di Ingegneria del Software
Alma Mater Studiorum · Università di Bologna

Indice

1 Descrizione del prodotto

Il prodotto da realizzare è un client Twitter in grado di fornire analisi e funzionalità specifiche per determinati argomenti.

1.1 Scope

Il backlog è stato partizionato in cinque epiche:

Epica	**Descrizione**
Raccolta e analisi di tweet	Prime funzionalità per ricercare e analizzare tweet
Scacchi	Partita a scacchi contro gli utenti di Twitter che scelgono a maggioranza la mossa
Reazione a catena	Raccolta dei tentativi degli utenti di Twitter di indovinare la *catena finale*
L'Eredità	Raccolta dei tentativi degli utenti di Twitter di indovinare la *ghigliottina*
Fantacitorio	Raccolta e visualizzazione dei punteggi del Fantacitorio e ricerca delle squadre degli altri utenti

Le user stories contenute in ciascuna epica sono descritte nella Sezione 2.

1.2 Casi d'uso

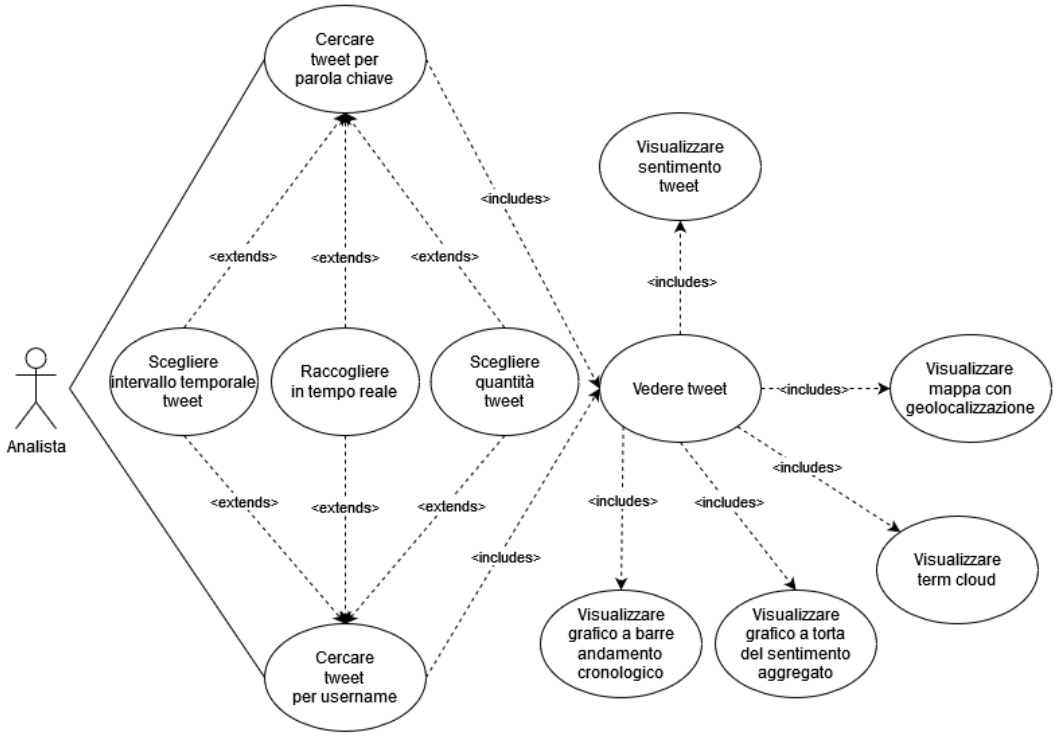


Figura 1: Casi d'uso per l'epica: Raccolta e analisi di tweet

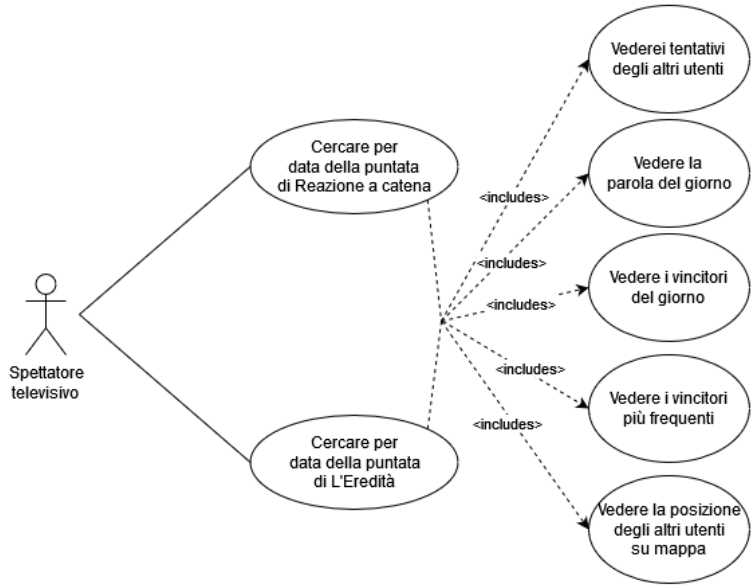


Figura 2: Casi d'uso per l'epica: Giochi televisivi

Figura 3: Casi d'uso per l'epica: Scacchi

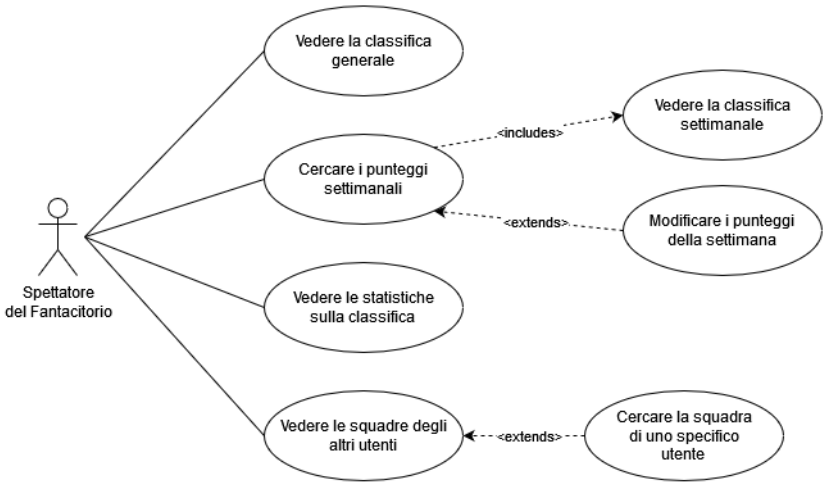


Figura 4: Casi d'uso per l'epica: Fantacitorio

1.3 Diagramma delle classi

Di seguito sono descritti, sottoforma di schede CRC, gli oggetti e le funzioni suddivisi per epica del backlog.
Per molte funzionalità del prodotto, si è ritenuto più adatto seguire un approccio funzionale (divisi in moduli) invece di utilizzare un approccio a oggetti.
Nonostante ciò, si è preferito descrivere la progettazione come schede CRC, anche se le entità non sono propriamente classi in quanto per ciascun modulo è possibile individuare le responsabilità e le dipendenze.

Raccolta e analisi di tweet

Nome: `fetch/keyword`	
Responsabilità • Raccolta di tweet per parola chiave	**Collaborazioni**

Nome: `fetch/user`	
Responsabilità • Raccolta di tweet per nome utente	**Collaborazioni**

Nome: `fetch/multiple_tweets`	
Responsabilità • Raccolta di un determinato numero (potenzialmente alto) di tweet	**Collaborazioni** • `fetch/keyword` • `fetch/user`

Nome: `fetch/stream`	
Responsabilità • Apertura di uno stream per la raccolta di tweet in tempo reale • Lettura dallo stream • Chiusura dello stream	**Collaborazioni**

Nome: `analysis/language`	
Responsabilità • Rilevazione della lingua di un testo	**Collaborazioni**

Nome: `analysis/sentiment`	
Responsabilità • Rilevazione del sentimento di un testo	**Collaborazioni** • `analysis/language`

Nome: `analysis/stopwords`

Responsabilità	**Collaborazioni**
• Rimozione di stop words da un testo	• `analysis/language`

Scacchi

Nome: `ChessOpponent`

Responsabilità	**Collaborazioni**
• Pubblicazione tweet con lo stato della scacchiera • Selezione della mossa successiva a maggioranza	• Modulo per l'autenticazione e l'invio di tweet

Nome: `ChessGame`

Responsabilità	**Collaborazioni**
• Tracciamento dello stato della partita • Rilevamento conclusione partita	

Reazione a catena e L'Eredità

Nome: `games/winningWord`

Responsabilità	**Collaborazioni**
• Estrazione della parola vincente	• `fetch/keyword`

Nome: `games/userAttempts`

Responsabilità	**Collaborazioni**
• Raccolta dei tentativi degli utenti	• `fetch/keyword`

Fantacitorio

Nome: `games/fantacitorio`

Responsabilità	**Collaborazioni**
• Raccolta dei punteggi settimanali • Calcolo della classifica generale • Raccolta delle squadre degli utenti • Calcolo di statistiche sulla classifica	• `fetch/keyword` • `fetch/user`

2 Descrizione degli sprint

Sono stati svolti quattro sprint della durata di 14 giorni ciascuno.
La stima dei punti delle user stories è stata effettuata con una scala da 0 a 10, valutando separatamente il frontend dal backend. Il punteggio complessivo è quindi ottenuto dalla somma di quest'ultimi.

2.1 Sprint 1

2.1.1 Sprint goal

Lo sprint è stato principalmente dedicato a studiare le API di Twitter e produrre le prime funzionalità per la visualizzazione e l'analisi dei tweet.
In particolare le feature pianificate per lo sprint sono state:

- Ricerca di tweet per username
- Ricerca di tweet per hashtag
- Analisi dei tweet tramite componenti grafiche:
  - Grafico a torta per il sentiment analysis
  - Grafico a barre per la frequenza dei Tweet
  - Word cloud con le parole più usate

2.1.2 Backlog

US: Come utente interessato ai tweet,
voglio poter cercare dei tweet per hashtag **Punti**: 8
per leggerli. (3 frontend + 5 backend)
Epica: Raccolta e analisi di tweet

DOD: L'utente, cercando un hashtag in un apposito textbox, è in grado di leggere tutti i tweet correlati visualizzando: nome account Twitter, username, immagine profilo, contenuto tweet (testo + foto e video), data e ora, luogo (se applicabile), numero like, numero commenti, numero retweet.
Test: Richiamare l'API implementata, verificare che il formato sia corretto e che il contenuti dei tweet contenga l'hashtag ricercato.

US: Come utente interessato ai tweet,
voglio poter cercare dei tweet per nome utente **Punti**: 8
per leggerli. (3 frontend + 5 backend)
Epica: Raccolta e analisi di tweet

DOD: L'utente, cercando un nome utente in un apposito textbox, è in grado di leggere tutti i tweet correlati visualizzando: nome account Twitter, username, immagine profilo, contenuto tweet (testo + foto e video), data e ora, luogo (se applicabile), numero like, numero commenti, numero retweet.
Test: Richiamare l'API implementata, verificare che il formato sia corretto e che l'autore dei tweet sia quello ricercato.

US: Come analista,
voglio poter analizzare il sentimento
per stabilire se un tweet è positivo o meno.
Punti: 9
(2 frontend + 7 backend)
Epica: Raccolta e analisi di tweet

DOD: L'utente, dato un tweet, vede se è positivo, negativo o neutro tramite immagine o testo.
Test: Analizzare frasi di cui è noto il sentimento.

US: Come analista,
voglio vedere un grafico a barre
per vedere il numero di tweet nell'unità di tempo.
Punti: 2
(2 frontend)
Epica: Raccolta e analisi di tweet

DOD: L'utente apre una pagina web contenente il grafico a barre con il numero di tweet nell'unità di tempo.
Test: Manualmente verificare che il grafico sia corretto.

US: Come analista,
voglio vedere un grafico a torta
per vedere il rapporto di sentiment positivi, negativi e neutri.
Punti: 2
(2 frontend)
Epica: Raccolta e analisi di tweet

DOD: L'utente apre una pagina web contenente il grafico a torta con sentiment positivi, negativi e neutri.
Test: Manualmente verificare che il grafico sia corretto.

US: Come analista,
voglio vedere una term cloud
per vedere le parole più utilizzate nei tweet.
Punti: 4
(2 frontend + 2 backend)
Epica: Raccolta e analisi di tweet

DOD: L'utente apre una pagina web contenente una term cloud con le parole più utilizzate.
Test: Manualmente verificare che il grafico sia corretto.

2.1.3 Esito sprint

Lo sprint è terminato con la conclusione di tutte le user stories pianificate.
Il lavoro è risultato omogeneo, con un piccolo scostamento rispetto all'andamento ideale che però non è stato causa di imprevisti o ritardi.

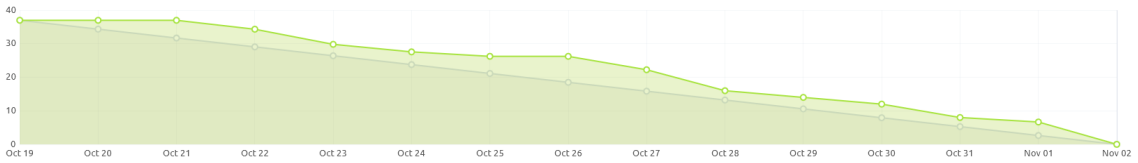


Figura 5: Burndown sprint 1

Anche le ore di lavoro (con il tempo stimato in quanto allo sprint 1 non è stato effettuato il monitoraggio) sono risultate in linea con il monte ore previsto.

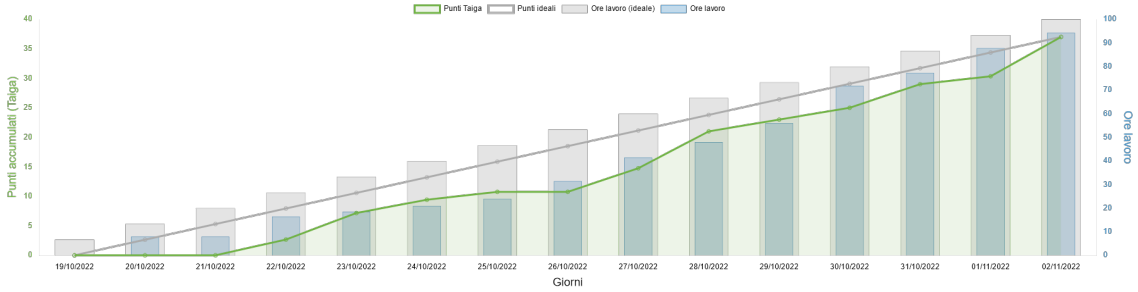


Figura 6: Progresso dei punti (asse a sinistra) e ore di lavoro (asse a destra)

2.1.4 Sprint review

Alla sprint review è stato chiesto dal cliente l'implementazione delle seguenti feature:

- Ricerca di tweet per intervallo temporale
- Possibilità di selezionare il numero di tweet da estrare con una singola richiesta

2.1.5 Retrospettiva

Pre-retrospettiva

Alla pre-retrospettiva effettuata a metà sprint, sono emerse le seguenti problematiche:

- Tempo dedicato al lavoro non sufficiente
- Mancanza di comunicazione nel team
- Task e user stories hanno descrizioni poco dettagliate e facilmente fraintendibili

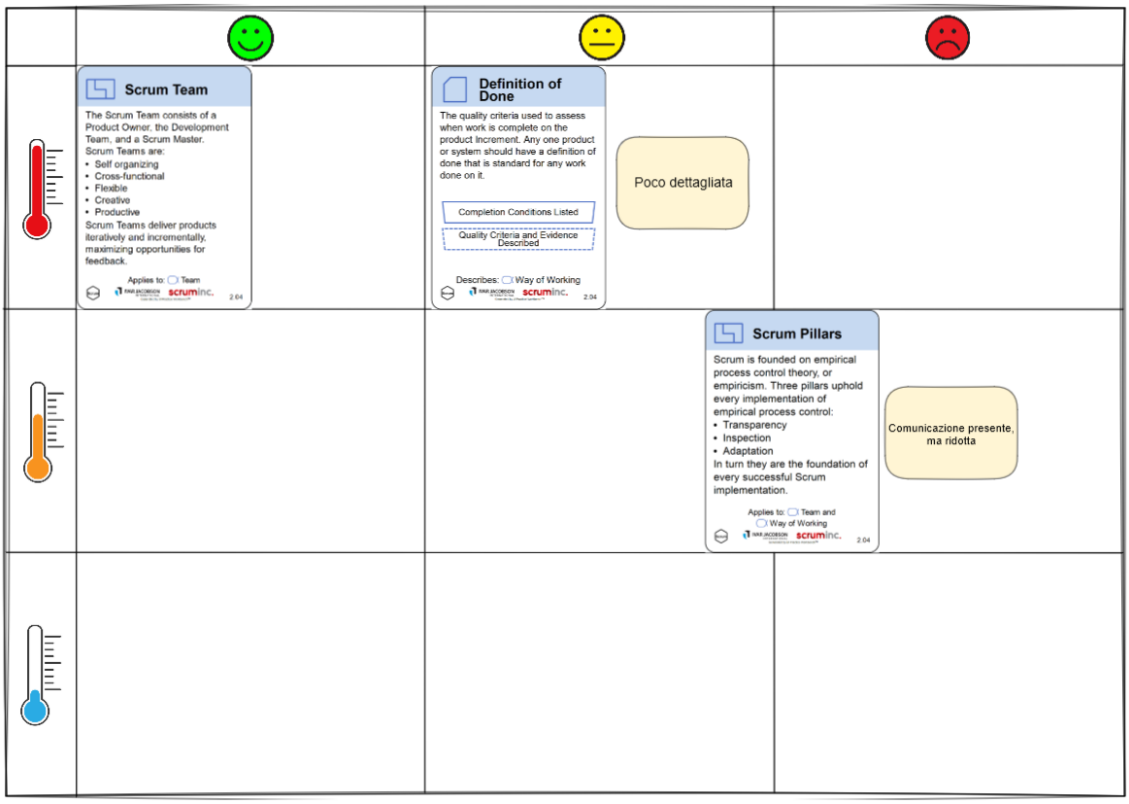


Figura 7: Pre-retrospettiva del 27/10/2022

Retrospettiva

Alla retrospettiva di fine sprint, gli aspetti positivi evidenziati dal team sono:

- Il deliverable presentato alla sprint review è risultato adeguato e funzionante
- Il team e i ruoli di ciascuno sono ben definiti

Sono invece stati portati all'attenzione del team i seguenti problemi:

- Non è stato effettuato un monitoraggio delle ore di lavoro (che è stato quindi stimato sulla base sui daily scrum virtuali)

- I problemi evidenziali alla pre-retrospettiva non sono ancora stati risolti

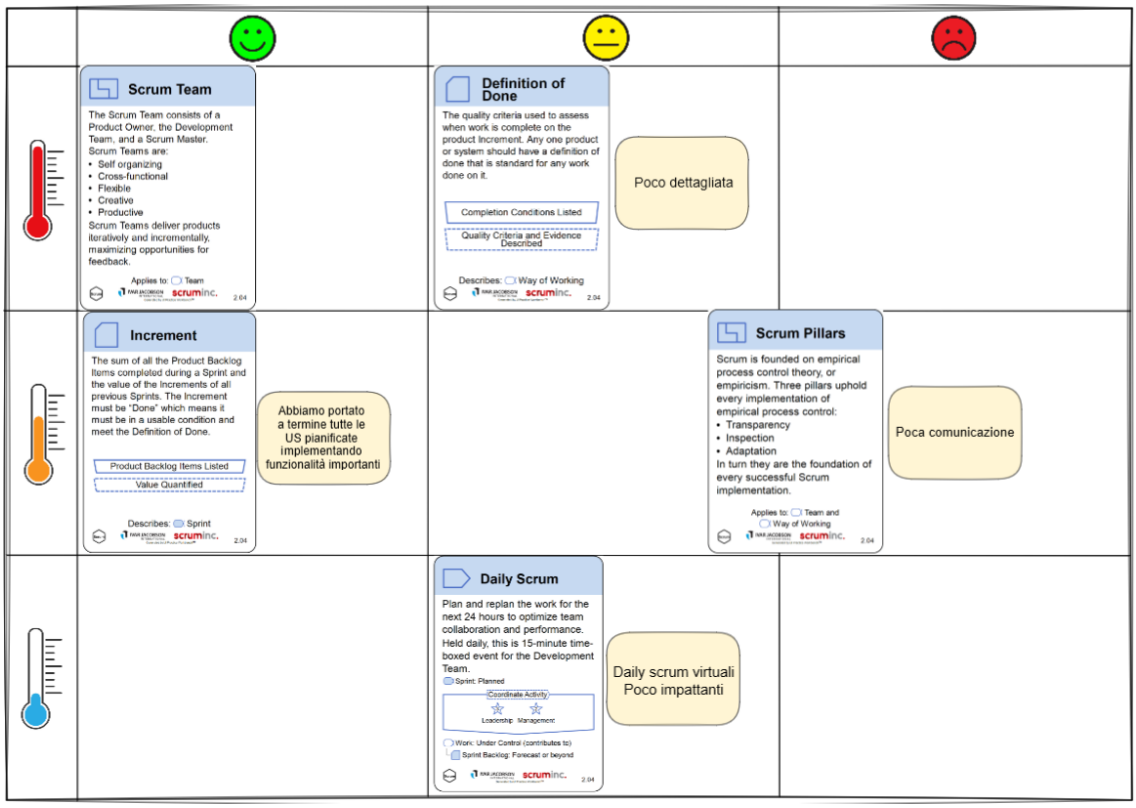


Figura 8: Retrospettiva del 01/11/2022

2.2 Sprint 2

2.2.1 Sprint goal

L'obiettivo dello sprint è stato quello di concludere l'epica riguardante la visualizzazione e l'analisi dei tweet.
Nello specifico, sono state implementate le seguenti funzionalità:

- Ricerca di tweet per intervallo temporale (richiesto dal cliente allo sprint review precedente)
- Ricerca di un determinato numero di tweet con una singola ricerca (richiesto dal cliente allo sprint review precedente)
- Ricerca di tweet per parola chiave
- Mappa per visualizzare la posizione dei tweet con geolocalizzazione
- Raccolta di tweet in tempo reale

2.2.2 Backlog

US: Come utente interessato a vedere tweet,
voglio scegliere un numero di tweet da poter caricare in una volta
per poterli analizzare in modo aggregato.
Epica: Raccolta e analisi di tweet

Punti: 5
(3 frontend + 2 backend)

DOD: Possibilità di ricercare un largo numero di tweet scrivendone la quantità in un textbox.
Tale quantità serve per la ricerca iniziale e anche per mostrare pagine successive. È possibile inoltre ricercare un numero di tweet inizialmente per poi cambiare quantità e cercare una pagina successiva (esempio: ricerca di 150 tweet iniziali, poi viene modificata la quantità (dallo stesso textbox) in 20 e si preme su "Pagina successiva". Il numero di tweet così mostrato diventa 170).
Test: Verificare che il numero di tweet raccolto corrisponda con quello richiesto.

US: Come utente interessato a vedere tweet,
voglio scegliere un intervallo di tempo in cui raccogliere tweet
per analizzarne le tendenze storiche.
Epica: Raccolta e analisi di tweet

Punti: 5
(3 frontend + 2 backend)

DOD: Possibilità di ricercare dei tweet dato un intervallo temporale. Non deve essere possibile cercare tweet nel futuro.
Non deve essere possibile cliccare su "Prossima pagina" quando non ci sono più tweet da visualizzare.
Test: Verificare che i tweet raccolti siano compresi nell'intervallo temporale.

US: Come utente interessato a vedere tweet,
voglio poter cercare dei tweet per parola chiave
per vedere cosa ne pensa la gente a riguardo.
Punti: 4
(2 frontend + 2 backend)
Epica: Raccolta e analisi di tweet

DOD: Possibilità di cercare tweet per parola o frase chiave. I grafici già presenti devono funzionare anche con questa ricerca.
Test: Richiamare l'API implementata, verificare che il formato sia corretto e che il contenuti dei tweet contenga la parola chiave ricercata.

US: Come utente interessato ai tweet,
voglio poter visualizzare su una mappa la posizione dei tweet cercati
per avere un'idea della località dalla quale sono stati pubblicati.
Punti: 7
(3 frontend + 4 backend)
Epica: Raccolta e analisi di tweet

DOD: Possibilità di visualizzare una mappa con le posizioni dei tweet ricercati.
Se sono presenti più tweet nella stessa zona è possibile aggregarli (in base alla distanza) e mostrare un unico valore, ovvero il numero di tweet in tale zona. La mappa deve essere sempre visibile anche durante lo scorrimento della pagina (come i grafici).
Test: Manualmente verificare che la mappa contenga i marker quando sono presenti tweet con geolocalizzazione.

US: Come utente,
voglio vedere la posizione di tutti i tweet di una data persona
per conoscere i suoi spostamenti.
Punti: 5
(5 frontend + 0 backend)
Epica: Raccolta e analisi di tweet

DOD: Possibilità di inserire il nome utente di una persona e visualizzare su una mappa le posizioni dei suoi tweet e i suoi spostamenti.
Per gli spostamenti si mostrano sulla mappa delle frecce basandosi sulla posizione e sulla data del tweet.
Test: Manualmente verificare che la mappa contenga i marker quando sono presenti tweet con geolocalizzazione.

US: Come lettore di tweet,
voglio poter vedere i tweet che ricerco in tempo reale
per sapere cosa la gente posta.
Punti: 10
(3 frontend + 7 backend)
Epica: Raccolta e analisi di tweet

DOD: Poter scrivere una ricerca e, al click di un pulsante "Live", vedere tutti i tweet pubblicati in tempo reale a partire da quel momento.
Test: Verificare che il socket implementato restituisca i tweet ricercati, quando disponibili.

2.2.3 Esito sprint

Lo sprint è terminato con la conclusione di tutte le user stories pianificate.
Il lavoro si è svolto in linea con l'andamento ideale dei punti. Si è osservato una leggera sovrastima delle user stories che ha portato alla conclusione anticipata (di un giorno) dello sprint.

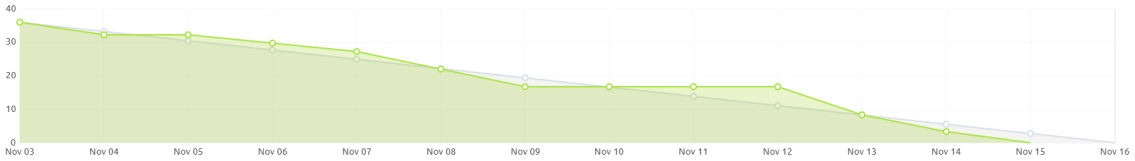


Figura 9: Burndown sprint 2

Le ore di lavoro rispettano il monte ore e sono state distribute principalmente in due periodi di maggiore produttività.

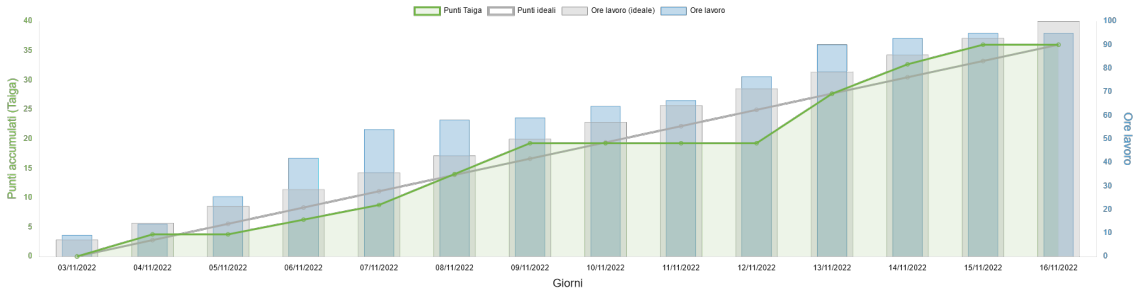


Figura 10: Progresso dei punti (asse a sinistra) e ore di lavoro (asse a destra)

2.2.4 Sprint review

Alla sprint review non sono emerse nuove richieste da parte del cliente.

2.2.5 Retrospettiva

Pre-retrospettiva

Alla pre-retrospettiva effettuata a metà sprint, sono emerse le seguenti problematiche:

- Mancanza di test adeguati per il frontend
- Le user stories sono state sovrastimate

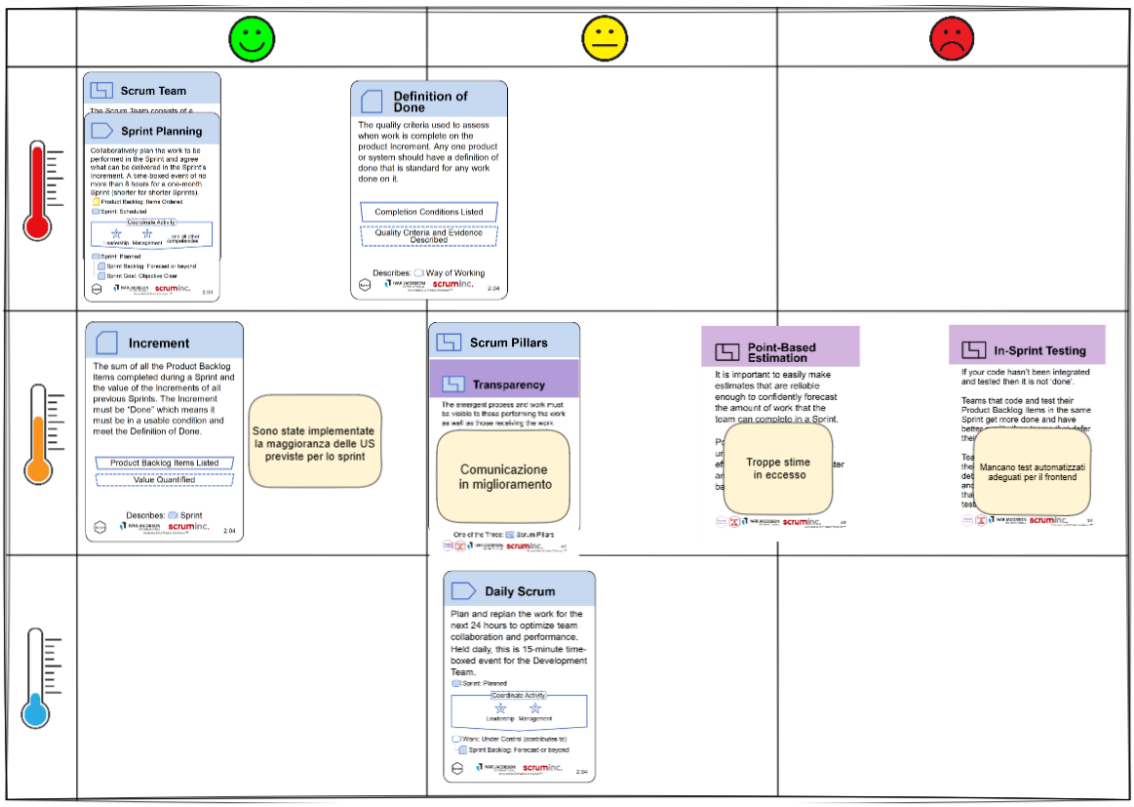


Figura 11: Pre-retrospettiva del 10/11/2022

Retrospettiva

Alla retrospettiva di fine sprint sono state confermate le problematiche della pre-retrospettiva.

2.3 Sprint 3

2.3.1 Sprint goal

Per lo sprint è stata pianificata l'implementazione delle epiche riguardanti i giochi televisivi (L'Eredità e Reazione a Catena) e l'inizio dell'epica riguardante lo scacchi.
In particolare sono state previste le seguenti funzionalità:

- Visualizzare gli utenti che tentano a indovinare la parola del giorno
- Visualizzare la parola vincente
- Visualizzare gli utenti che indovinano la parola
- Visualizzare la posizione degli utenti che tentano di indovinare
- Possibilità di muovere le proprie pedine su una scacchiera

2.3.2 Backlog

US: Come spettatore de #leredita,
voglio raccogliere i tweet di chi prova a indovinare la ghigliottina,
per visualizzare, in ordine temporale, tutti coloro che provano ad indovinare.
Epica: L'Eredità

Punti: 5
(2 frontend + 3 backend)

DOD: Possibilità di visualizzare una pagina con tutti i tweet di tutte le persone che usano #leredita e provano ad indovinare la ghigliottina.
Test: Cercando per data, verificare che vengano visualizzati i tentativi del giorno.

US: Come spettatore de #leredita,
voglio visualizzare, su una mappa, la posizione di tutti coloro che provano ad indovinare
per conoscere la posizione dei giocatori.
Epica: L'Eredità

Punti: 4
(4 frontend)

DOD: Possibilità di visualizzare una pagina con tutte le posizioni su una mappa dei tweet di tutte le persone che usano #leredita e provano ad indovinare la ghigliottina.
Test: Manualmente verificare che nella mappa siano presenti i marker dei tweet con la geolocalizzazione.

US: Come spettatore de #leredita,
voglio visualizzare tutti coloro che indovinano la ghigliottina
per sapere chi ha indovinato.
Epica: L'Eredità

Punti: 7
(4 frontend + 3 backend)

DOD: Possibilità di visualizzare la parola del giorno assieme a tutti i vincitori che hanno indovinato.
Test: Cercando per data, verificare che venga trovata la parola del giorno.

US: Come spettatore di #reazioneacatena,
voglio raccogliere i tweet di chi prova a indovinare
l'ultima parola,
per visualizzare, in ordine temporale, tutti coloro che
provano ad indovinare.
Epica: Reazione a catena

Punti: 2
(2 backend)

DOD: Possibilità di visualizzare una pagina con tutti i tweet di tutte le persone che usano #reazioneacatena e provano ad indovinare la parola finale.
Test: Cercando per data, verificare che vengano visualizzati i tentativi del giorno.

US: Come spettatore di #reazioneacatena,
voglio visualizzare, su una mappa, la posizione di tutti
coloro che provano ad indovinare
per conoscere la posizione di giocatori.
Epica: Reazione a catena

Punti: 0
Segue dalla US de #leredita

DOD: Possibilità di visualizzare una pagina con tutte le posizioni su una mappa di tweet di tutte le persone che usano #reazioneacatena e provano ad indovinare la parola finale.
Test: Manualmente verificare che nella mappa siano presenti i marker dei tweet con la geolocalizzazione.

US: Come spettatore de #reazioneacatena,
voglio visualizzare tutti coloro che indovinano l'ultima
parola
per sapere chi ha indovinato.
Epica: Reazione a catena

Punti: 2
(2 backend)

DOD: Possibilità di visualizzare la parola del giorno assieme a tutti i vincitori che hanno indovinato.
Test: Cercando per data, verificare che venga trovata la parola del giorno.

US: Come giocatore di scacchi,
voglio poter muovere una pedina
per fare la mia mossa.
Epica: Scacchi

Punti: 12
(6 frontend + 6 backend)

DOD: Possibilità di fare una mossa a scacchi e visualizzarla.
Test: Provare a effettuare mosse valide e invalide.

2.3.3 Esito sprint

Lo sprint è terminato con la conclusione di tutte le user stories pianificate.

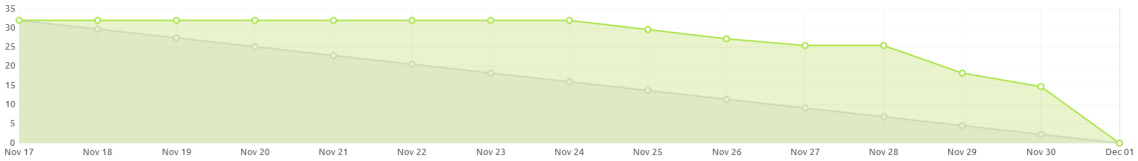


Figura 12: Burndown sprint 3

Il lavoro è stato però distribuito in maniera disomogenea, infatti, nella prima metà dello sprint non sono stati fatti progressi significativi, mentre tutto il valore è stato portato nella seconda metà dello sprint. Conseguentemente anche le ore di lavoro sono tutte distribuite nel secondo periodo dello sprint.
Il monte ore "standard" non è stato raggiunto, ma ciò era atteso in quanto durante la lo sprint planning era già stata prevista una minore quantità di lavoro.

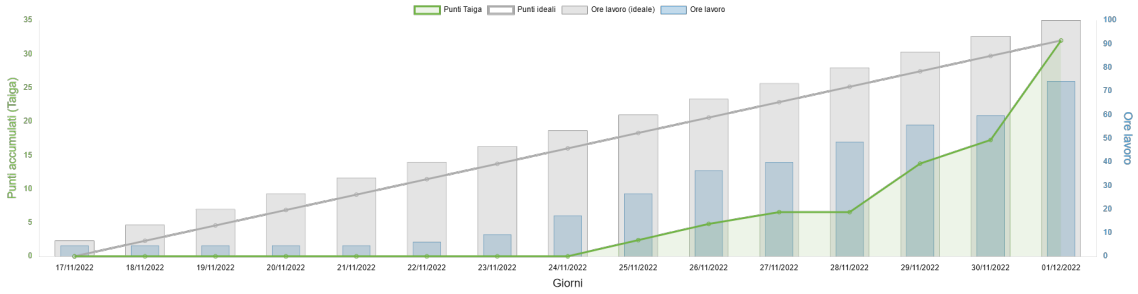


Figura 13: Progresso dei punti (asse a sinistra) e ore di lavoro (asse a destra)

2.3.4 Sprint review

Alla sprint review il cliente ha espresso le seguenti richieste:

- Revisione del grafico a barre della frequenza dei tweet, soprattutto in situazioni in cui sono presenti solo tweet di una giornata
- Aggiungere ai giochi televisivi la visualizzazione dei giocatori più vincenti in un periodo di tempo

Inoltre è stata aggiunta alle specifiche, con priorità massima, l'implementazione di funzionalità riguardanti il Fantacitorio.

2.3.5 Retrospettiva

Dalla retrospettiva sono emerse le seguenti problematiche:

- Abbiamo sottovalutato lo sprint e lavorato poco di conseguenza
- La DoD di alcune user stories risultavano ambigue

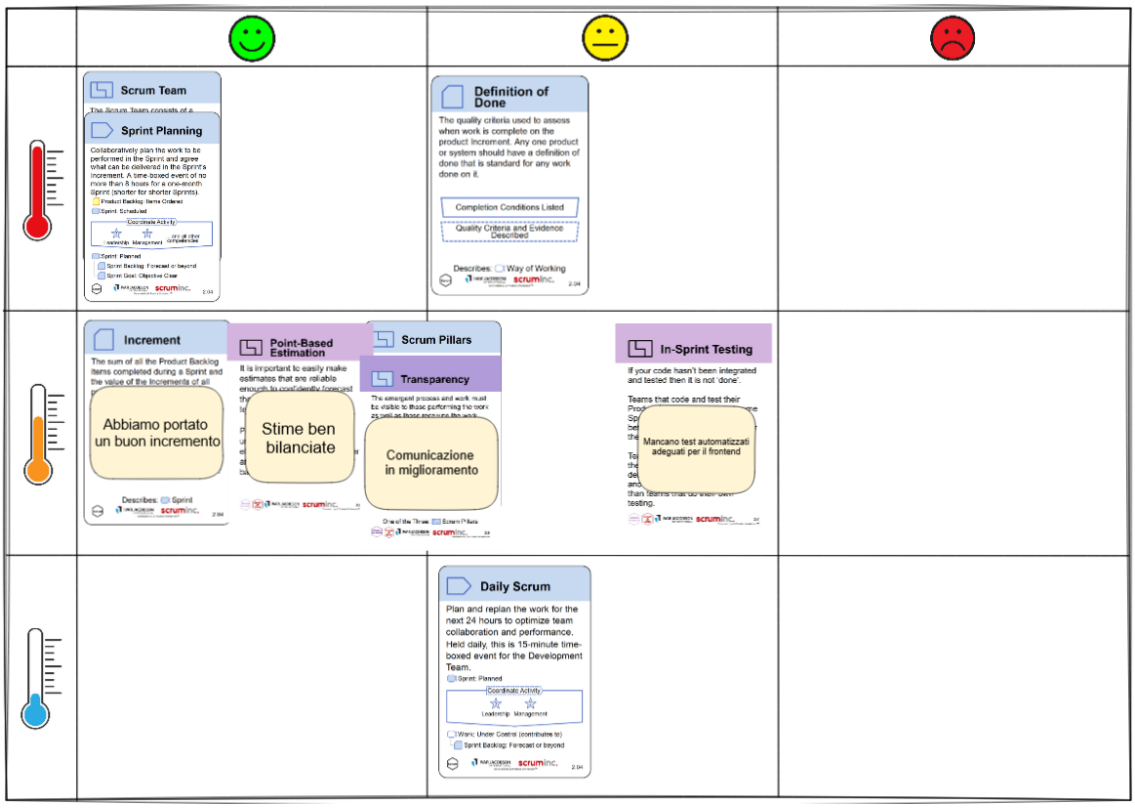


Figura 14: Retrospettiva del 02/12/2022

2.4 Sprint 4

2.4.1 Sprint goal

L'obiettivo dello sprint finale è stato quello di implementare il Fantacitorio (richiesto allo sprint precedente), concludere lo scacchi e integrare le richieste del cliente emerse dalla sprint review precedente nel prodotto finale.
In particolare sono state previste le seguenti funzionalità:

- Per il Fantacitorio:
  - Visualizzazione e modifica della classifica settimanale
  - Visualizzazione della classifica generale
  - Visualizzazione delle squadre degli altri giocatori
  - Ricerca della squadra di uno specifico utente
  - Statistiche sulla classifica
- Per lo scacchi:
  - Pubblicazione della scacchiera come tweet
  - Raccolta e selezione della mossa dell'avversario a maggioranza
- Per i giochi televisivi:
  - Visualizzazione dei più vincenti in un periodo di tempo

2.4.2 Backlog

US: Come interessato al Fantacitorio,
voglio vedere e poter aggiornare una sintesi settimanale dei punteggi dei politici
per vedere chi sta vincendo.
Epica: Fantacitorio

Punti: 11
(4 frontend + 7 backend)

DOD: Avere una lista di tutti i politici con i loro relativi punteggi, visualizzarne una sintesi e poter modificarla.
Test: Verificare il corretto parsing dei tweet contentente i punteggi.

US: Come interessato al Fantacitorio,
voglio vedere una classifica cumulativa di punteggi di ogni politico
per capire chi sta vincendo.
Epica: Fantacitorio

Punti: 5
(2 frontend + 3 backend)

DOD: Visualizzare una classifica cumulativa con i punti di ogni politico in ordine decrescente.
Test: Verificare che la classifica sia coerente con i punteggi settimanali.

US: Come interessato al Fantacitorio,
voglio sfogliare le immagini delle squadre degli altri partecipanti
per vedere contro chi competo.
Epica: Fantacitorio

Punti: 7
(3 frontend + 4 backend)

DOD: Visualizzare, in una pagina, tutte le foto delle squadre partecipanti al Fantacitorio.
Test: Verificare manualmente che l'algoritmo sia in grado di riconoscere correttamente le immagini contenente una squadra.

US: Come interessato al Fantacitorio,
voglio poter cercare un utente e vedere se partecipa o meno e mostrare la sua squadra
per cercare chi gioca.
Epica: Fantacitorio

Punti: 5
(2 frontend + 3 backend)

DOD: Possibilità di cercare un utente per username e controllare se possiede una squadra, in caso affermativo mostrarla.
Test: Verificare che cercando un utente che partecipa al Fantacitorio, la sua squadra sia visibile.

US: Come giocatore di scacchi,
voglio che la scacchiera venga pubblicata
in modo tale che gli utenti vedano la mossa scelta.
Epica: Scacchi

Punti: 3
(3 backend)

DOD: Pubblicare un tweet con la foto della scacchiera allo stato attuale.
Test: Verificare manualmente che il tweet sia stato pubblicato.

US: Come giocatore di scacchi,
voglio che gli utenti di Twitter scelgano, a maggioranza, la mossa dell'avversario
per avanzare nella partita.
Epica: Scacchi

Punti: 4
(1 frontend + 3 backend)

DOD: Scelta di una mossa in base alla maggioranza dei voti presenti nei commenti del post pubblicato.
Test: Verificare che la mossa scelta sia quella proposta dal maggior numero di utenti.

US: Come spettatore de #leredita,
voglio visualizzare colui/colei che ha indovinato più volte nel corso delle puntate,
per sapere chi è più bravo/a.
Epica: L'Eredità

Punti: 5
(2 frontend + 3 backend)

DOD: Possibilità di visualizzare, giorno per giorno, una classifica con le persone con più parole indovinate.
Test: Verificare che gli utenti visualizzati abbiano effettivamente vinto il numero di volte indicato.

US: Come spettatore de #reazioneacatena,
voglio visualizzare colui/colei che ha indovinato più volte nel corso delle puntate,
per sapere chi è più bravo/a.
Epica: Reazione a catena

Punti: 1
(1 backend)

DOD: Possibilità di visualizzare, giorno per giorno, una classifica con le persone con più parole indovinate.
Test: Verificare che gli utenti visualizzati abbiano effettivamente vinto il numero di volte indicato.

US: Come interessato al Fantacitorio,
voglio visualizzare delle statistiche interessanti nella classifica
per vedere chi sta andando bene e chi no.
Epica: Fantacitorio

Punti: 3
(1 frontend + 2 backend)

DOD: Mostrare statistiche interessanti nella pagina della classifica, come ad esempio "best climber", "best average" e "best single score".
Test: Verificare che le statistiche siano in linea con la classifica.

2.4.3 Esito sprint

Lo sprint è terminato con la conclusione di tutte le user stories pianificate.

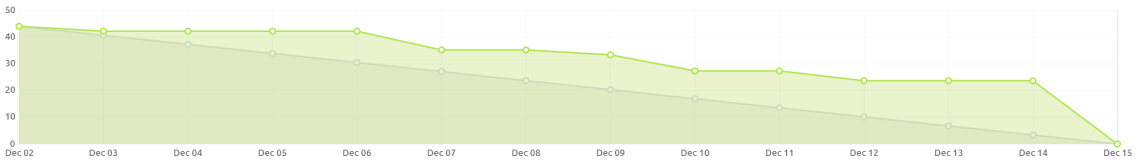


Figura 15: Burndown sprint 4

La distribuzione delle ore di lavoro è risultata disomogenea, con un'importante concentrazione all'ultimo giorno dello sprint.

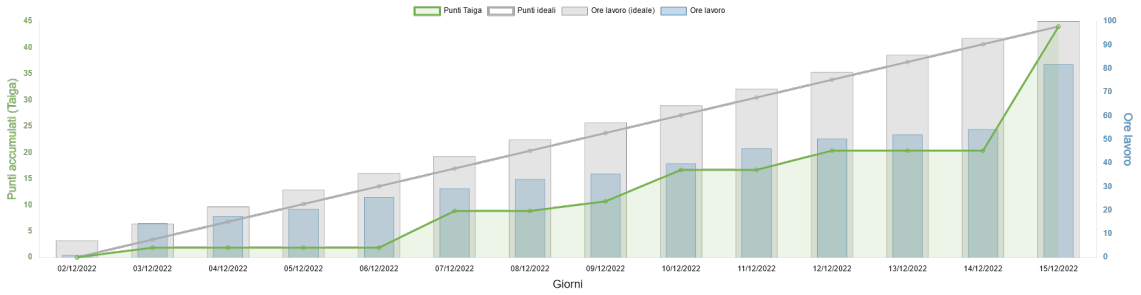


Figura 16: Progresso dei punti (asse a sinistra) e ore di lavoro (asse a destra)

2.4.4 Retrospettiva

Alla retrospettiva il team ha evidenziato i seguenti aspetti positivi:

- Una buona reazione alle modifiche del backlog
- Un incremento significativo nei test (del frontend)

È stato invece evidenziato come aspetto negativo un mal bilanciamento del lavoro, concentrato tutto alla fine dello sprint.

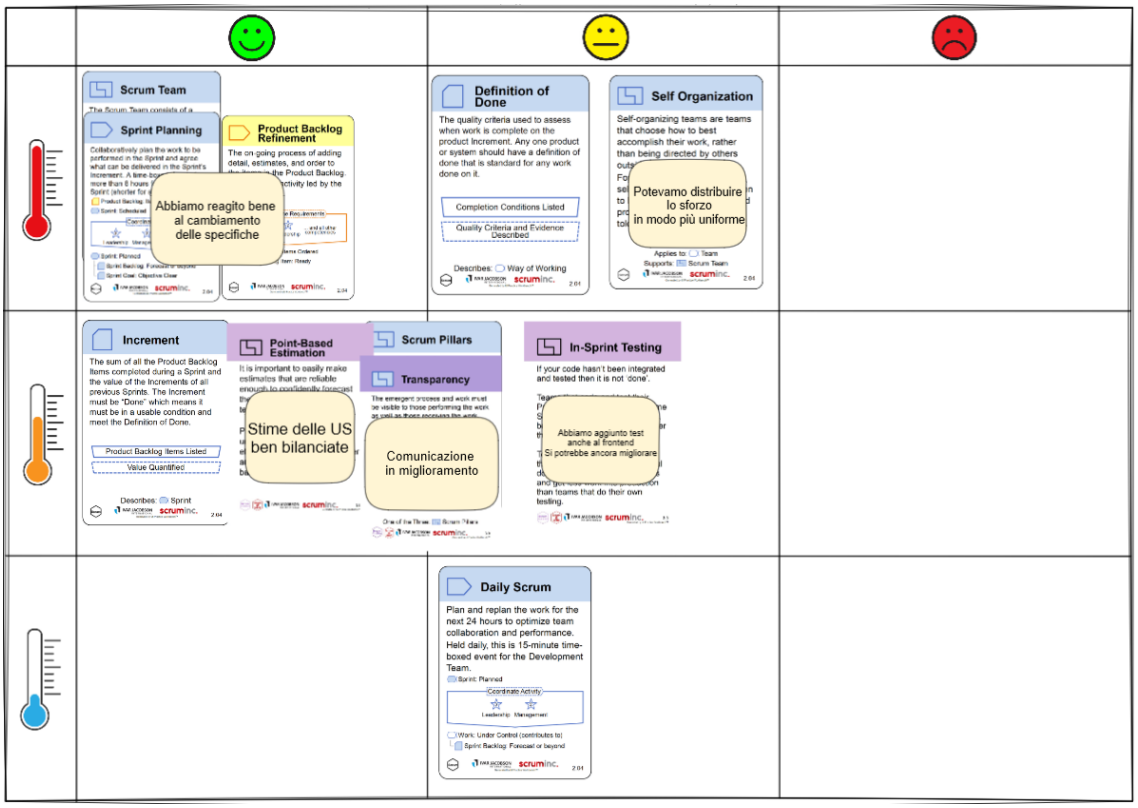


Figura 17: Retrospettiva del 02/12/2022

3 Descrizione del processo

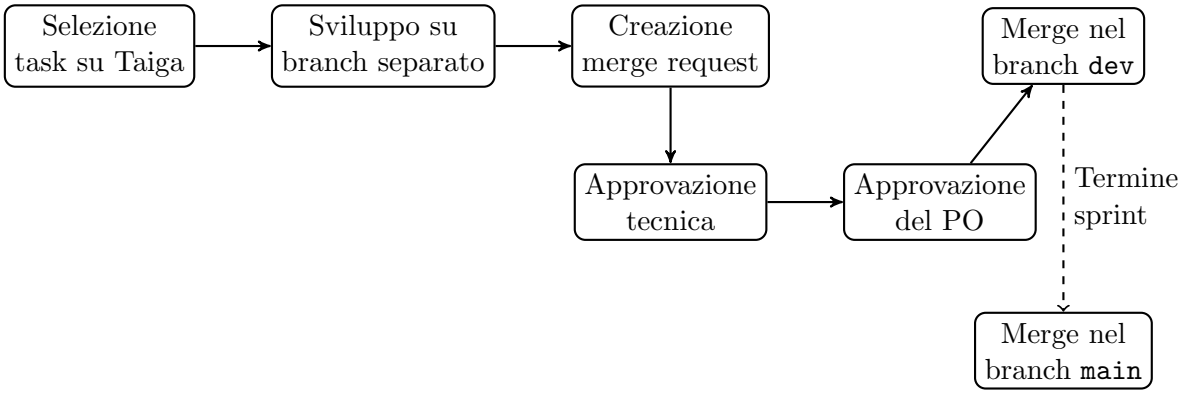
3.1 Flusso di lavoro

Le responsabilità di ciascun componente del gruppo nel processo di sviluppo sono stati i seguenti[1]:

Membro	**Responsabilità**
Cheikh Ibrahim Zaid PO Operativo	Sviluppo backend. Testing e quality assurance.
Lee Qun Hao Henry Developer	Sviluppo frontend.
Paris Manuel Developer	Sviluppo backend.
Xia Tian Cheng Scrum master	Sviluppo fullstack. Sistemista e CI/CD.

3.1.1 Git

L'organizzazione del repository git segue la metodologia *gitflow* con un branch principale (`main`) che dopo ogni sprint viene allineato con il branch di sviluppo (`dev`).
Più nello specifico, il flusso del lavoro ha seguito il seguente schema:



3.1.2 Bug tracking

Per il tracciamento di problemi e bug sono stati usati gli strumenti disponibili in Gitlab e Taiga. In particolare è stato utilizzato la funzionalità *Issues* di Gitlab per creare e segnalare bug. Contemporaneamente è stato impostato Taiga in modo tale che venga sincronizzato con Gitlab.
In questo modo sono visibile su entrambi gli strumenti le segnalazioni e il loro stato, permettendo una migliore pianificazione in fase di sprint planning e una maggiore organizzazione durante lo sprint.

[1] Ad eccezione dello sprint 3 durante il quale è stato sperimentato l'inversione dei ruoli del team di sviluppo

3.1.3 Daily scrum

Data l'impossibilità di effettuare incontri quotidiani, i daily scrum sono stati realizzati con cadenza più irregolare, tenendo una riunione almeno una volta a settimana per allineare il team alla situazione del progetto.
In aggiunta, con l'obiettivo sopperire alla mancanza di incontri giornalieri, sono stati effettuati dei *daily scrum virtuali* su Mattermost. Nello specifico, quando un componente decide di lavorare in una giornata, scrive un messaggio contenente la struttura di un daily scrum (ciò che svolgerà, ciò che ha svolto l'ultima volta ed eventuali problemi) in modo tale da informare il team dell'avanzamento dei lavori.

3.1.4 Monitoraggio delle ore di lavoro

Per il tracciamento delle ore di lavoro è stata adottata una soluzione "manuale". Ogni componente del gruppo è responsabile del monitoraggio delle proprie ore che compila in un file `json` presente in un repository su Gitlab[2].
Nello stesso repository è presente una webapp[3] che elabora le informazioni inserite e si interfaccia con le API di Taiga, producendo una sintesi dei dati assieme a qualche statistica.

3.2 Team building

3.2.1 Scrumble

La partita a Scrumble è stata giocata allo sprint 0, durante la fase preparatoria del progetto.
Lo scopo della sessione di team building era quella di prendere maggiore confidenza con la metodologia Scrum e confermare i ruoli all'interno del gruppo (Autovalutazione [1]).

3.2.2 Escape the Boom

La partita a Scrumble è stata giocata allo sprint 1.
L'obiettivo era quello di rafforzare la comunicazione nel team, che, come segnalato dal team alla retrospettiva (Sezione 2.2.5), era carente e poco significativa (Autovalutazione [2]).

3.3 Gitinspector

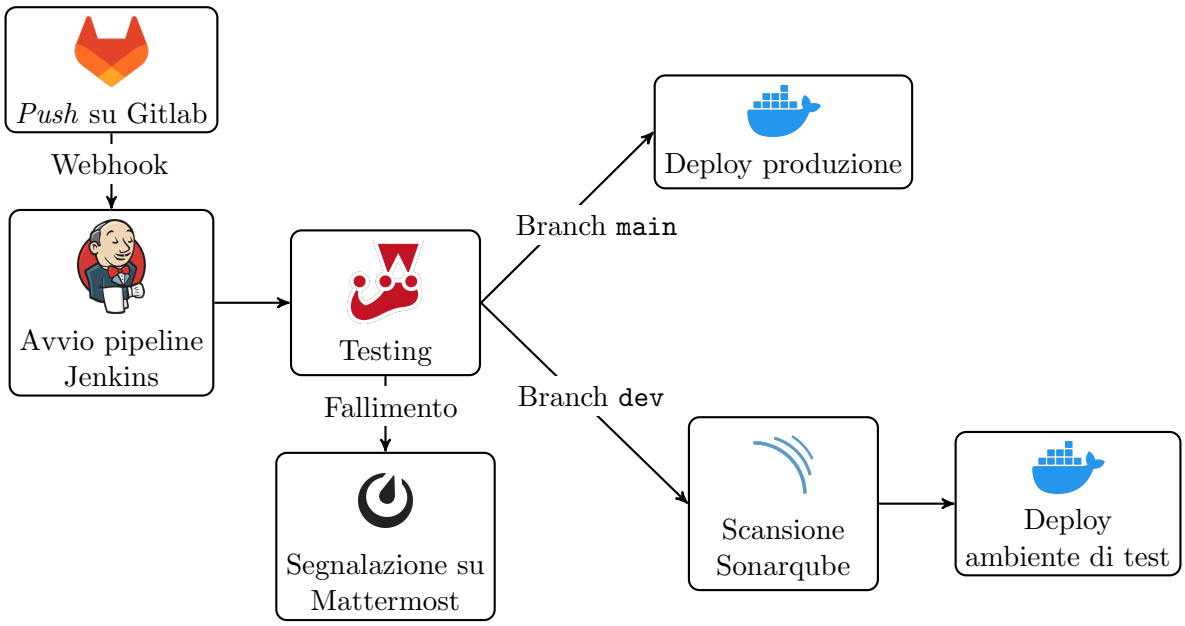
I risultati di Gitinspector sono disponibili nel repository Gitlab [3]. Per un'analisi più gradulare, sono state fatte le seguenti scansioni:

- Complessiva su tutti i file
- Solo sui file che implementano funzionalità (quindi escludendo i test)
- Solo sui file contenenti test

[2]Repository per le ore di lavoro: `https://tcxia.ddns.net/gitlab/t12/time-tracker`
[3]Webapp con sintesi delle ore di lavoro: `https://tcxia.ddns.net/time/`

3.4 Deployment

Il seguente schema rappresenta le fasi della pipeline definita per il progetto:



Sonarqube è definito solo sul branch di sviluppo (`dev`) in quanto la versione *community* è limitata alla scansione di un unico branch.
Il deploy avviene su container Docker allineati al codice del branch di riferimento. Il database risiede su un container separato.

4 Artefatti

[1] Team 12. GQM Scrumble. `Download da Taiga` oppure `Link al wiki su Taiga`. 15 ottobre 2022.

[2] Team 12. GQM Escape the Boom. `Download da Taiga` oppure `Link al wiki su Taiga`. 30 ottobre 2022.

[3] Tian Cheng Xia. Statistiche generate da Gitinspector. `Link al repository Gitlab`.