Gestione del Progetto UniMeet

Versione 1.0.0 del 20/11/25

Versione	Modifiche
1.0.0	Compilazione

Bregolin Gabriele
Leggeri Leonardo
Vecchi Samuele

1. Ciclo di Vita del Software

1.1 Modello di processo adottato

Per lo sviluppo del progetto è stato adottato un modello di ciclo di vita iterativo e incrementale, implementato attraverso il framework Agile Scrum.

La scelta di Scrum è stata dettata dalla necessità di gestire requisiti in evoluzione e di avere un feedback frequente sulle funzionalità sviluppate. Il ciclo di vita è stato suddiviso in iterazioni temporali fisse (Sprint) con cadenza settimanale.

Il team ha ricoperto i ruoli chiave previsti dal framework (Product Owner, Scrum Master, Development Team) senza una separazione netta delle responsabilità tra i membri, gestendo il lavoro tramite il Product Backlog e la Kanban Board.

1.2 Organizzazione temporale e deviazioni dal piano

L'organizzazione temporale ha seguito una struttura a Sprint, con una durata media prevista di 1 settimana. Tuttavia, rispetto al piano ideale, il processo ha subito un adattamento dovuto ai vincoli del calendario accademico.

Cronologia degli Sprint:

- **Sprint 1:** [20/11/2025 - 27/11/2025] - Analisi del dominio e modellazione del sito
    - Focus:Definire il perimetro del progetto e validare l'architettura tecnica iniziale.
- **Sprint 2:** [28/11/2025 - 07/12/2025] - Sicurezza e persistenza dei dati
    - Focus: Implementare un sistema di accesso sicuro e garantire la continuità dei dati.
- **Sprint 3:** [8/12/2025 - 16/12/2025] - Logica delle sessioni di studio e interfaccia grafica
    - Focus: implementare in modo corretto la logica delle sessioni e la parte grafica di quest'ultime.
- **Sospensione Tecnica:** [17/12/2025 - 13/01/2026]
    - *Nota:* In questo periodo lo sviluppo attivo è stato sospeso per permettere ai membri del team di gestire priorità accademiche esterne al progetto.
- **Sprint 4:** [14/01/2026 - 21/01/2026] - Ruoli amministrativi, sistema di notifiche, area personale e visibilità pubblica dei profili
    - Focus: Differenziare l'accesso al sito tra utenti e amministratori e gestire la comunicazione asincrona dei messaggi fra gli utenti. Realizzare le dashboard utente e consentire la visualizzazione dei profili altrui.
- **Sprint 5:** [26/01/2026 - 30/01/2026] - Logica Social e Networking tra Colleghi
    - Focus: Implementare il sistema di relazioni tra utenti e la gestione delle richieste di "following".
- **Sprint 6:** [31/01/2026 - 3/02/2026] - Logica Social e Networking tra Colleghi
    - Focus: Implementare il sistema di relazioni tra utenti e la gestione delle richieste di "following".

- **Sprint 7 / Finalizzazione:** [3/02/2026 - 5/02/2026] - Ottimizzazione, Refactoring e Validazione Finale
    - Consolidare l'intero sistema, ottimizzare le performance del database e correggere i bug UI.

Dopo la sospensione tecnica del 17 dicembre - 13 gennaio è stato effettuato uno sprint retrospective per riallineare il team e valutare quali sarebbero dovuti essere i prossimi passi.

1.3 Software Product Line

Non è stato ritenuto necessario adottare un approccio SPL in quanto il prodotto non prevede varianti strutturali significative.

2. Gestione della Configurazione

2.1 Strumenti e Ambiente

Per la gestione della configurazione e il versionamento del codice sorgente è stato utilizzato il sistema di controllo distribuito **Git**, con hosting remoto sulla piattaforma **GitHub**.

La scelta di GitHub ha permesso di centralizzare non solo il codice, ma anche la gestione dei task e il tracciamento delle problematiche, integrando il ciclo di vita dello sviluppo in un unico ambiente.

2.2 Workflow Operativo

Il team ha adottato un flusso di lavoro basato su **Feature Branching**. Il ramo main è stato mantenuto sempre stabile e distribuibile.

Le operazioni quotidiane dei membri del team hanno seguito la seguente sequenza di comandi standard:

1. Aggiornamento del repository locale: git pull origin main
2. Creazione di un nuovo branch per il task specifico: git checkout -b feature/nome-feature
3. Esecuzione delle modifiche e staging dei file: git add .
4. Commit delle modifiche con messaggio descrittivo: git commit -m "Descrizione modifica"
5. Invio delle modifiche al repository remoto: git push origin feature/nome-feature

2.3 Gestione Issues e Project Board (Kanban)

Per la pianificazione e l'assegnazione dei task è stata utilizzata la **Kanban Board**. Ogni requisito o bug è stato convertito in una **Issue** su GitHub. La board è stata organizzata nelle seguenti colonne per riflettere lo stato di avanzamento:

- **Backlog:** Issues pianificate ma non ancora iniziate.

- **Ready:** Issue pianificate per lo spring corrente ma non ancora iniziate.
- **In Progress:** Issue attualmente in sviluppo.
- **Review:** Issue sviluppate in attesa di approvazione
- **Done:** Issue completate, tramite Pull Request, uniti al branch principale.

Questo approccio visuale ha permesso di monitorare il *Work In Progress* (WIP) e identificare eventuali colli di bottiglia.

2.4 Branching, Pull Requests e Code Review

Per garantire l'integrità del codice nel branch main, è stata imposta una policy che vieta i commit di feature pesanti direttamente sul ramo principale. Ogni integrazione è avvenuta tramite **Pull Request (PR)**.

Il processo di **Code Review** è stato implementato come passaggio obbligatorio prima del merge:

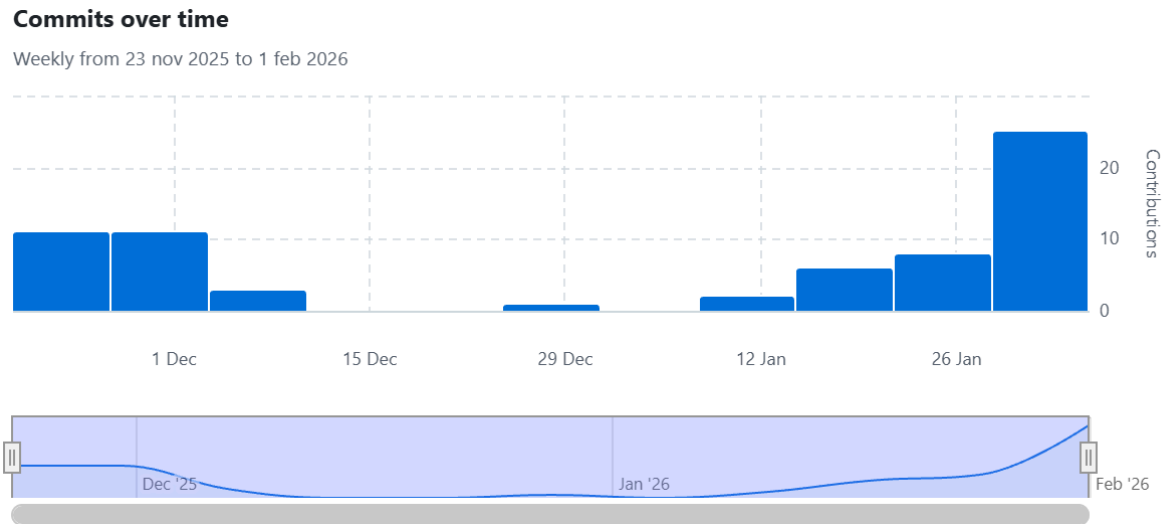
- Una volta completato lo sviluppo su un branch secondario, lo sviluppatore apre una PR verso il main.
- La PR funge da spazio di revisione: i membri del team verificano che le modifiche non introducano conflitti o regressioni.
- Solo dopo la verifica (approvazione), la PR viene chiusa e il codice unito.

2.5 Statistiche del Repository (Documentazione)

A testimonianza dell'attività svolta durante il ciclo di vita del progetto, si riportano i seguenti dati quantitativi estratti da GitHub al termine dello sviluppo:

2.5.1 Total Commits

Sono stati effettuati 31 commit (escludendo i merge), sono stati modificati 102 file, sono state aggiunte 10.722 righe e rimosse 3.510 righe.



2.5.2 Issues Closed

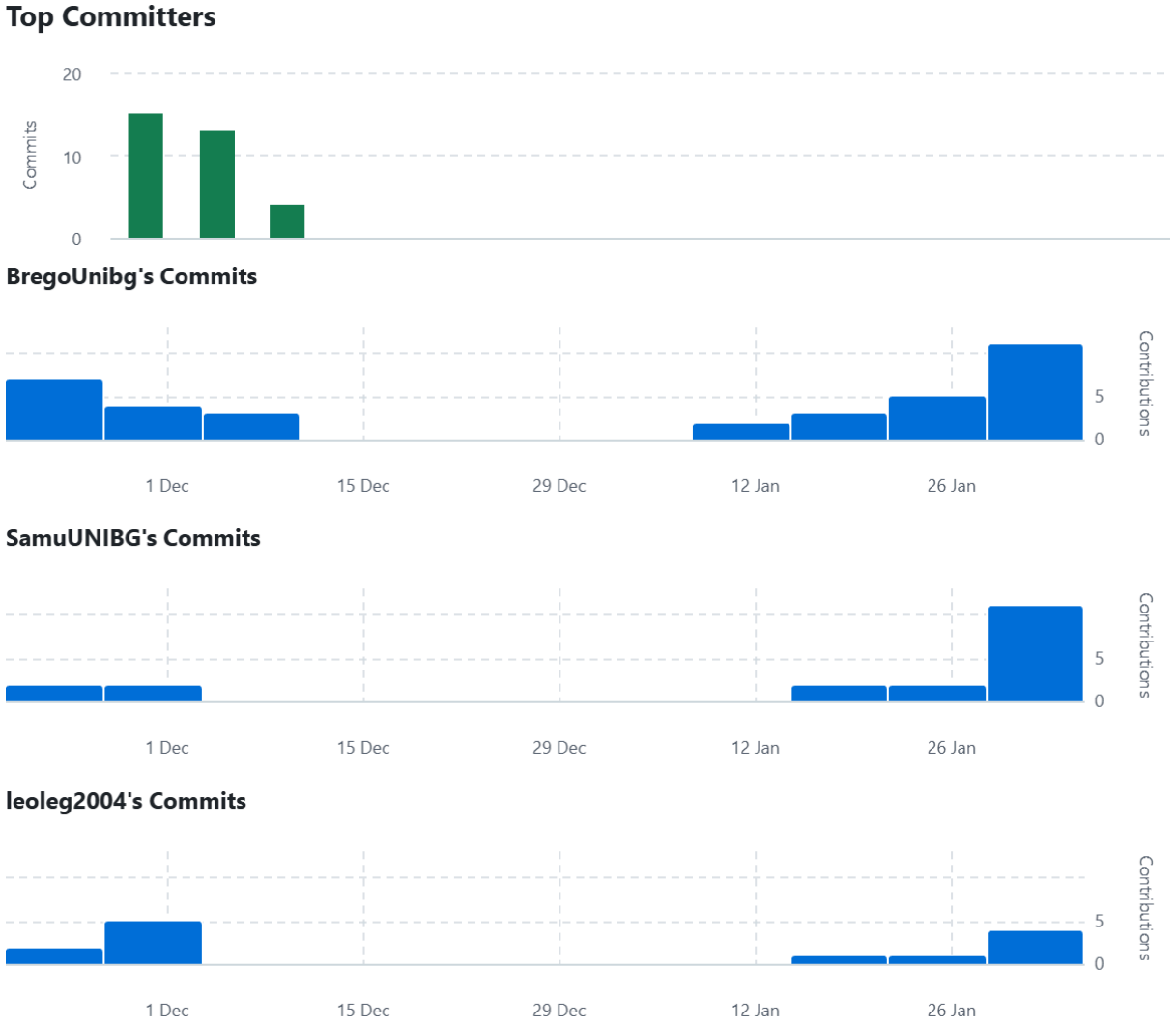
Sono state chiuse 20 issue.

2.5.3 Pull Requests

20 integrazioni effettuate.

2.5.4 Contributors

All'implementazione del codice hanno collaborato 3 persone .



3. Organizzazione del Team e People Management

3.1 Composizione del Team

Il team di sviluppo è costituito dai seguenti membri:

- **Bregolin Gabriele**
- **Leggeri Leonardo**
- **Vecchi Samuele**

3.2 Modello Organizzativo (Riferimento Teorico)

In conformità con la natura accademica del progetto e le dimensioni ridotte del gruppo, è stata adottata una struttura organizzativa di tipo **Democratico decentrato**.

In questo modello:

- **Assenza di gerarchia:** Non è stato designato un Project Manager autoritario. La leadership è stata condivisa e basata sul consenso.
- **Comunicazione totale:** La struttura di comunicazione è "molti-a-molti", dove ogni membro ha interagito direttamente con tutti gli altri per coordinare le attività.
- **Decision Making:** Le decisioni critiche (architetturali, tecnologiche e di design) sono state discusse collettivamente durante le riunioni di pianificazione.

3.3 Gestione dei Ruoli e Responsabilità

Coerentemente con il framework Scrum adottato, il team ha operato come un'unità **Cross-Funzionale** e **Auto-Organizzata**. Invece di assegnare ruoli rigidi e verticali, si è optato per un approccio dinamico basato sul concetto di *Collective Code Ownership*:

- **Development Team:** Tutti i membri hanno contribuito alla scrittura del codice sia Backend che Frontend, garantendo che la conoscenza del sistema fosse condivisa e non confinata a un singolo individuo.
- **Ruoli Scrum:**
    - Il ruolo di **Product Owner** è stato simulato collettivamente dal team in fase di analisi dei requisiti.
    - Il ruolo di **Scrum Master** è stato gestito a rotazione o in modo diffuso, assicurandosi reciprocamente che gli ostacoli venissero rimossi.

3.4 Meccanismi di Comunicazione e Coordinamento

Per gestire il lavoro distribuito e mantenere l'allineamento, sono stati utilizzati i seguenti canali:

- **Sincrono:** Incontri settimanali su Discord per lo *Sprint Planning* e la revisione del lavoro.
- **Asincrono:** Utilizzo di un gruppo WhatsApp per comunicazioni rapide e urgenti.

- **Collaborativo:** Commenti sulle Issue di GitHub per discussioni tecniche legate a specifici task.