Project Plan UniMeet

Versione 0.1.0 del 14/11/25

Versione	Modifiche
0.1.0	Compilazione project plan

Bregolin Gabriele
Leggeri Leonardo
Vecchi Samuele

Indice

1. Introduzione

UniMeet è una piattaforma di organizzazione pensata per gli studenti universitari, il cui scopo è facilitare ed incentivare la creazione di gruppi di studio.
Gli studenti, ovvero gli utenti della applicazione, avranno la possibilità di creare sessioni di studio visibili ai colleghi, quest'ultimi se interessati potranno richiedere di partecipare.
Il progetto verrà svolto da: Bregolin Gabriele, Leggeri Leonardo, Vecchi Samuele.

2. Modello di Processo

Per questo progetto è stato scelto il modello di processo AGILE Scrum.
In via del tutto eccezionale essendo il team composto da solo 3 membri con pari capacità decisionali ognuno di essi sarà Product Owner, avrà quindi facoltà di decidere cosa includere nel Product Backlog (ossia il documento di requirement engineering ove verranno esplicitate funzionalità ed elementi del prodotto finale).

Gli Sprint avverranno su cadenza settimanale. A capo di ogni Daily Scrum vi sarà uno solo Scrum Master e verrà fatto il punto della situazione riguardo a progressi, possibili disguidi e issues aperti.

Alla fine di ogni Sprint il team farà una breve analisi qualitativa e quantitativa in merito ai risultati raggiunti, per poi pianificare l'iterazione successiva (Sprint Review & Retrospective).

Di seguito le pietre miliari individuabili per l'applicazione:

- Possibilità di inserire e modificare nuove strutture e aule
- Possibilità di inserire e modificare sessioni di studio
- Possibilità di richiedere di partecipare a sessioni di studio presenti nel sistema

Verrà verificato il raggiungimento di ciascuna pietra miliare tramite testing approfondito, successivamente automatizzato per accertare la continua funzionalità durante lo sviluppo.
Punti critici ai quali prestare particolare attenzione saranno l'implementazione del Database Server embedded.

3. Organizzazione del progetto

Il progetto è voluto dal team di sviluppo stesso. Quest'ultimo è autonomo in quanto padrone di se stesso, non risponde ad alcun ente terzo.
Il Team non dispone di una gerarchia, piuttosto di un'organizzazione piatta e paritaria, ogni cambiamento della visione o dei requisiti del progetto del singolo membro verrà discusso con il resto del team e servirà l'approvazione informale da parte di tutti per la sua realizzazione; si hanno quindi pari responsabilità.
Lacune: per quanto riguarda l'implementazione del Database embedded e lo sviluppo dell'interfaccia il team non dispone di esperienze a posteriori, acquisire e conoscere i relativi strumenti sarà un processo di apprendimento critico.

4. Standard, linee guida, procedure

Nel corso di questo progetto verranno seguiti da tutti i membri del team standard e linee guida volte a dare la percezione che l'intero progetto sia stato svolto da una singola persona. In questo modo si migliora la fluidità e la chiarezza della documentazione e la testabilità e manutenibilità del codice.
Inoltre ci saranno procedure definite per la creazione e pubblicazione su github delle modifiche apportate a qualsiasi parte del progetto e per la segnalazione di eventuali errori riscontrati e da risolvere.

La documentazione dovrà quindi essere redatta evitando l'uso dei pronomi personali, adottando un registro impersonale. Per la struttura del contenuto si adotterà uno standard Business Letter: Arial, titoli (20pt), intestazioni (16pt, grassetto), testo (11pt).

Ogni documento sarà così definito:

- Prima pagina: titolo e versione
- Seconda pagina: indice
- A seguire: il contenuto del documento

Nel piè di pagina verrà riportato il numero della pagina.

L'intero codice java dovrà seguire le convenzioni definite da Oracle.

La pubblicazione su github del lavoro svolto dai singoli membri avverrà tramite push e commit. Ad ogni commit dovrà essere associato un commento sintetico che descriva le modifiche apportate al sistema.

Ogni membro, prima di iniziare a lavorare sul progetto, dovrà scaricare la versione funzionante più recente disponibile su github.
Per effettuare modifiche o progredire nell'implementazione di un requisito, sarà necessario creare un nuovo branch su github e pubblicare le modifiche apportate su quest'ultimo. Solo dopo aver verificato e validato il lavoro svolto, il branch creato potrà essere fuso con il branch principale.
Il nome del branch sarà in relazione allo sviluppo in corso su esso, per esempio l'implementazione dell'interfaccia di login potrebbe chiamarsi: feature-login; mentre la correzione di un errore potrebbe chiamarsi: bugfix-errorelogin.

Ogni volta che si apportano delle modifiche a qualsiasi parte del sistema, esse dovranno essere pubblicate su github entro fine giornata. Questo serve per garantire che ogni membro del team lavori sempre sulla versione più aggiornata delle componenti del software che si sta realizzando.

Ogni errore o problema riscontrato dovrà essere segnalato tramite l'apertura di un issue, contenente la descrizione del problema rilevato, dovrà essere assegnato al membro del team responsabile della correzione. Una volta risolto, l'issue viene chiuso.

5. Attività di Gestione

I requisiti saranno classificati in ordine di priorità in base al loro contributo per poter arrivare a una versione per lo meno essenziale ma perfettamente funzionante e completa del sistema.

Essi verranno quindi implementati in base al loro ordine di importanza: a partire dagli essenziali fino ad arrivare a quelli più estetici o opzionali.

Per monitorare lo stato di avanzamento del progetto si utilizzerà una Kanban Board, una lavagna dinamica per gestire il flusso di lavoro, costituita da 5 colonne dedite a tracciare lo stato di avanzamento dello sviluppo delle feature: backlog, prossimo alla realizzazione (scrum planning), in corso, accettazione/testing, completato.

6. Rischi

Potenziali rischi che si possono riscontrare nell'ambito di questo progetto riguardano incompetenza riguardo l'implementazione dell'embedded database server e lo sviluppo dell'interfaccia web, questo potrebbe portare a dei ritardi su tutto il resto del progetto.
Il livello di conoscenza degli strumenti e dei linguaggi utilizzati non è il medesimo per i membri del team, durante l'apprendimento delle funzionalità dei tool a disposizione il disallineamento delle conoscenze potrebbe ulteriormente ampliarsi.

7. Personale

Il personale comprende esclusivamente i membri del team menzionati in precedenza. Le conoscenze dei membri non sono né simmetriche né tantomeno complete, si punterà infatti sull'apprendimento in parallelo allo sviluppo. Per questo il team non assegna ruoli rigidi specializzati, piuttosto punta sulle conoscenze trasversali e sul continuo scambio reciproco dei compiti.

8. Metodi e tecniche

Essendo il team autonomo, quindi padrone di se stesso, dovrà portare in vita una sua visione piuttosto che adattare un sistema già esistente alle ultime tecnologie informatiche. L'obiettivo di questo progetto è lo sviluppo di una piattaforma unica, in quanto non ne esistono di simili, sarà quindi necessario astrarre dalla realtà sociale degli studenti delle attività da poter rappresentare ed organizzare.
Detto questo le tecniche adottare per l'elicitazione dei requisiti saranno:
Analisi basata sullo scenario e Descrizione in linguaggio naturale, quindi descrivere il comportamento degli studenti e modellizzarlo in modo tale da emularlo nel sistema.

In questa fase saranno prodotti tre diversi diagrammi UML: use case diagram, activity diagram e state machine diagram.

Durante la fase di progettazione verranno realizzati altri diagrammi UML, come: class diagram, sequence diagram, component diagram, communication diagram/timing diagram, e package diagram.

L’implementazione avverrà basandosi sui diagrammi prodotti nelle fasi precedenti, in particolare class diagram e sequence diagram.

Per quanto riguarda il sistema di controllo di versione verrà utilizzato Github, le formalità da seguire sono state definite in precedenza.

Infine, i test saranno svolti per verificare che i requisiti del sistema possono essere eseguiti correttamente e "gradevolmente", quindi con semplicità in maniere intuitiva, dall’utente finale (lo studente).

I componenti software saranno integrati e testati nel seguente ordine: interfaccia utente, database, web server.

9. Garanzia della qualità

Per definire e valutare la qualità del software che si andrà a sviluppare ci si baserà sul modello di McCall che identifica tre macro categorie: funzionamento del prodotto, manutenibilità del prodotto, transizione del prodotto.

Per garantire il rispetto di tali criteri si dovranno seguire i prossimi punti:

- Assicurarsi che il codice scritto non sia dipendente da uno specifico componente hardware o sistema operativo
- Assicurarsi che il codice abba poche dipendenze interne cicliche tramite l’impiego di uno specifico tool **(inserire il nome del tool usato dal prof, forse stanide)**
- Assicurarsi che nel codice non ci siano ripetizioni di metodi e funzionalità e che sia semplice (Complessità ciclomatica di McCabe)
- Assicurarsi che l’interfaccia utente sia user friendly (intuitiva)
- Monitorare l’utilizzo delle risorse di sistema durante le varie fasi di test
- Assicurarsi che tutti i casi di test siano implementati correttamente e che il sistema faccia ciò per cui è stato creato
- Assicurarsi che la risposta del sistema riguardo al suo utilizzo e testing rimanga consistente durante tutto il periodo di sviluppo.

portabilità: codice scritto non strettamente legato ad una specifica piattaforma o os

riusabilità: codice scritto con poche dipendenze (tramite software usato dal prof)

flessibilità e manutenibilità e testabilità: codice semplice senza ripetizioni

semplicità: interfaccia utente intuitiva

efficienza: monitoraggio durante il test delle risorse utilizzate

correttezza: implementazioni di tutti i casi di test

affidabilità: i singoli casi di test verranno eseguiti più e più volte e si verificherà la coerenza dei risultati

10. Risorse

Il sistema sarà composto da un semplice e leggero database embedded Java e web server integrato, quindi relativamente leggero e stabile, eseguibile dalla maggior parte dei laptop moderni. Qualora il progetto diventasse pubblico ed utilizzato, sarebbe necessario disporre di un server professionale sempre operativo e spazi di archiviazione significativi.
Per quanto riguarda le risorse software, quindi strumenti e librerie incluse, a priori sono stati identificati come candidati per rispettivamente il web server & framework e il database integrato: 2H e Vaadin.

11. Budget

Questo progetto non dispone di un budget in quanto tale. L'unica vera risorsa è il tempo a disposizione dell'organico, ovvero il tempo del team di sviluppo.

12. Cambiamenti

Per questo progetto è stato scelto un metodo di sviluppo AGILE per diverse cause: team piccolo, giovane ed inesperto, realtà modellata nuova e senza particolari vincoli, progetto scalabile e ricco di possibili aggiunte.
Il cambiamento o comunque l'aggiunta di obiettivi, quindi la modifica dei requisiti è prevista essere molto comune e verrà gestita in maniera informale.
Qualora un membro del team proponga una modifica dei requisiti o comunque di una funzionalità questa verrà discussa in una breve sessione di brainstorming, possibilmente durante il daily scrum. Se gradita e quindi approvata da tutti i membri si procederà con la modifica dei documenti di requirement engineering, quindi l'aggiunta del componente nel Product Backlog e nella Kanban Board, in attesa della sua implementazione.

13. Consegna

Tutta la documentazione e il codice sorgente del sistema verranno caricati su Github, ove sarà possibile vedere l'avanzamento e il progresso del lavoro fino ad arrivare alla versione finale.
Il team si pone come deadline il mese di dicembre, entro il primo gennaio dovrebbe quindi essere disponibile nella repository del progetto tutta la documentazione e l'applicazione completa.