M.E.D. Medical Environment Database

Project Plan

1. Introduzione

Il progetto nasce dalla volontà di sviluppare una soluzione software alternativa volta alla gestione automatica (raccolta, scrittura e consultazione) dei dati che accompagnano il paziente dalla sua accettazione alla dimissione contribuendo a raccogliere una serie di rilevazioni sui parametri vitali, al fine di migliorare l'efficienza ospedaliera e il mantenimento dei dati, il tutto con un'interfaccia semplice e intuitiva.

Il personale accede tramite log-in e successivamente filtra i degenti inserendo dati del paziente, grado di urgenza e reparto. I medici del pronto soccorso compilano una diaria medica durante la prima visita, modificabile in seguito. Dopo la compilazione, il paziente è da prendere in carico da parte degli operatori nei reparti di destinazione. Gli infermieri, assegnati a moduli specifici, compilano diarie infermieristiche e rilevazioni. I medici monitorano lo storico per adattare la cura. Al termine della degenza, il medico svolge la dimissione che elimina il paziente dal database.

L'idea di questo progetto ha avuto origine tramite il coinvolgimento in un dialogo con un nostro contatto operante nel contesto ospedaliero. La nostra conoscenza, divenuta il nostro "cliente", ci ha permesso di comprendere al meglio la realtà in esame. Lo sviluppo software prevede l'utilizzo del linguaggio java attraverso l'IDEE "Eclipse" e GitHub come spazio condiviso atto alla collaborazione.

2. Modello di processo

Il modello di processo selezionato è "SCRUM". La scelta di implementare un modello "agile" è stata motivata dalla dimensione contenuta del team di sviluppo e dall'assenza di una suddivisione rigida delle mansioni.

Le principali fasi di progettazione individuate come "Milestone" sono le seguenti:

- Definizione generale dell'aspetto grafico dell'applicazione
- Realizzazione del codice del Log-In Personale
- Completamento del modulo "Pronto Soccorso" tramite realizzazione di liste di pazienti
- Completamento del modulo "In attesa"
- Completamento dello scheletro del modulo "in reparto"
- Implementazione dei sistemi di farmaci + info
- Implementazione dei sistemi di diaria
- Implementazione dei sistemi di rilevazione
- Implementazione del sistema di dimissione

Dato il modello "SCRUM", al completamento di ognuna di queste milestone, che rappresentano il raggiungimento di un obiettivo significativo, si è deciso di effettuare una riunione di retrospezione straordinaria in aggiunta alle classiche riunioni post-sprint delle fasi assegnate dallo "SCRUM master".

3. Organizzazione del progetto

La configurazione del team di sviluppo adotta il paradigma della "Squadra Agile", conforme al modello di progetto precedentemente adottato. La suddivisione dei ruoli è stata concordata secondo le seguenti specifiche:

- Scrum Master: Gabriele Mazzoleni
- Product Owner: Filippo Antonio Bolis
- Sviluppatori, Progettisti, Tester: Gabriele Mazzoleni, Filippo Antonio Bolis, Daniele Gotti, Gabriele Masinari

In considerazione delle dimensioni contenute del team, si è deciso di assegnare a tutti i membri i ruoli di sviluppatori, progettisti e tester. Per quanto riguarda il Product Owner, è stato scelto il membro con maggiore interazione con il cliente e una conoscenza più approfondita del contesto di riferimento. Il ruolo di Scrum Master è stato affidato al componente del team che si assume la responsabilità di coordinare le attività, effettuando la selezione dei moduli del programma dallo sprint backlog e garantendo l'osservanza delle scadenze stabilite. Alla conclusione di ciascuno sprint, l'intero team si impegna a riunirsi per una sessione di retrospettiva, durante la quale viene condotto un test generale del modulo sviluppato.

4. Standard, linee guida e procedure

Il progetto è sviluppato mediante il linguaggio di programmazione Java, all'interno dell'ambiente di sviluppo Eclipse. Ci atteniamo alle convenzioni standard di codifica di Java, che includono la corretta nomenclatura per classi, metodi, variabili e l'adozione di pratiche di indentazione adeguate. Al fine di gestire le dipendenze del progetto, utilizziamo il sistema di gestione dei progetti Maven.

Per quanto riguarda la creazione dell'interfaccia grafica utente (GUI), utilizziamo il plugin WindowBuilder. Inoltre, per automatizzare la documentazione del codice Java, si fa ricorso a JAutoDoc, mentre per la gestione query il framework Jooq. La creazione e gestione del database e delle relative tabelle sono affidate a SQL Lite. Infine per i Test usiamo JUnit e per le metriche abbiamo utilizzato Structure101 Studio.

5. Attivita di gestione

In linea con il modello selezionato, la gestione e il controllo dei progressi avvengono alla fine di ogni giornata lavorativa, attraverso una riunione o un incontro informale per fare il punto della situazione. Nel corso dello sprint, in caso di necessità, si aprono degli "Issues" la cui risoluzione viene assegnata a uno sviluppatore dallo Scrum Master e convalidata dall'intero team. Al termine dello sprint si esegue una valutazione collettiva più approfondita insieme a un test delle funzionalità. Al completamento di ogni milestone il Product Owner si confronta con il cliente.

6. Rischi

Il software non pone il problema della persistenza del dato in caso che questo non sia gestito correttamente da un gestore di database.

Nel caso il Database Manager assunto decida di implementare un database cloud, il software non cripta i dati inviati tra database e client, non impedendo a terzi non autorizzati di accedere con conseguenti rischi legali.

Essendo la prima versione pubblicata, potrebbe non essere esente da bug dovuti a utilizzi imprevisti o impropri del programma.

Non essendo l'applicazione collegata ad un database di farmaci in mercato, il sistema non previene potenziali errori di battitura nell'inserimento del farmaco durante la prescrizione.

7. Personale

1079011 - Gotti Daniele

1079493 - Bolis Filippo Antonio

1079514 - Mazzoleni Gabriele

1079692 - Masinari Gabriele

8. Metodi e tecniche

La gestione delle future versioni o degli aggiornamenti del software viene eseguita successivamente al rilascio, in risposta a eventuali errori o modifiche richieste dal cliente. Durante le attività di manutenzione del software, possono essere implementate nuove funzionalità o apportate correzioni per ottimizzare ulteriormente il prodotto.

Nel corso della fase di collaudo, il cliente utilizza il software per verificare il soddisfacimento dei requisiti e il corretto funzionamento di tutte le sue funzioni. In caso di necessità, vengono prese in considerazione ulteriori richieste che potranno essere implementate nelle versioni successive.

La documentazione viene prodotta al termine di ogni milestone. Qualora si renda necessario l'utilizzo di nuovi pacchetti o strumenti per proseguire lo sviluppo del software, la documentazione viene aggiornata.

Dopo il completamento di una milestone, si procede alla fase di testing. Un esito positivo del test comporta l'implementazione delle relative componenti nel progetto.

Componenti software: Eclipse IDE, GitHub/GitHub Desktop, DB Browser, Draw.io, StarUML, Structure101.

Pacchetti: WindowBuilder, JAutoDoc, Jooq, SQLLite, JUnit.

9. Garanzia di qualità

Al completamento di ogni milestone ogni componente del gruppo effettua un controllo del codice validato poi dallo Scrum Master. La fase di testing e la successiva verifica da parte del cliente consentiranno di garantire i requisiti di qualità.

10. Pacchetti di lavoro

Si prevede di suddividere lo sviluppo delle milestone in fasi documentate nello sprint backlog. L'attività da eseguire settimanalmente è selezionata dallo Scrum Master e tutta la squadra si occupa di portare a termine il compito suddividendosi i ruoli in base alla necessità.

11. Risorse

Uno o più personal computer con sistema operativo Windows 10/ Windows 11, conformi ai requisiti minimi di sistema per utilizzare Eclipse IDE e DB Browser.

12. Budget

Poiché si tratta di un progetto privo di spese finanziarie, l'unico criterio per la pianificazione del budget è il tempo. In particolare, si prevede un impegno di lavoro di 60 ore per ciascun membro del gruppo.

13. Cambiamenti

In conformità con il modello adottato, ciascuna modifica alle pianificazioni operative, sia riguardante il product backlog sia di altra natura, sarà approvata dallo Scrum Master e poi caricata su GitHub.

Sono stati adottati ulteriori cambiamenti all'architettura del software, in quanto si è deciso di implementare un ulteriore pacchetto di lavoro chiamato model, passando così da una architettura a tre livelli a una variante del modello M-V-C

14. Consegna

Si prevede di consegnare l'intero progetto entro il termine del mese di gennaio 2024.