

Progetto ingegneria del software M.E.D(Medical Environment Database)

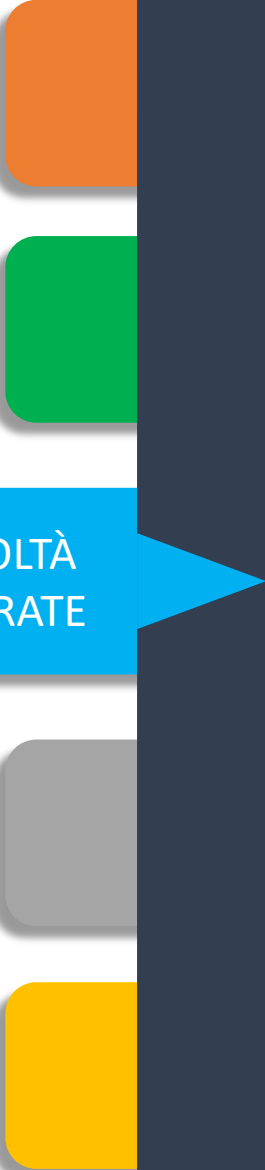


- **Gotti Daniele**, matricola 1079011
- **Bolis Filippo Antonio**, matricola 1079493
- **Mazzoleni Gabriele**, matricola 1079514
- **Masinari Gabriele**, matricola 1079692



OBIETTIVO

L'obiettivo del progetto è di sviluppare un software alternativo finalizzato alla gestione automatica dei dati relativi al percorso clinico del paziente, compresi raccolta, scrittura e consultazione. Questa soluzione mira a raccogliere informazioni sui parametri vitali al fine di migliorare l'efficienza ospedaliera e la gestione dei dati. Il software è progettato con un'interfaccia semplice e intuitiva per facilitare l'utilizzo durante tutto il processo, gestendo il percorso clinico del paziente dal suo arrivo fino alla sua dimissione.



Utilizzo di Github(inizialmente)
Realizzazione del diagramma delle classi
//definire

DIFFICOLTÀ
INCONTRATE



Programmazione: Programmazione ad oggetti
Modellazione: Diagrammi UML



Java, Markdown, SQL



Eclipse IDE , StarUML, Github/Github Desktop,
draw.io, Window Builder, Db browser.



GitHub



Issues, branches, pull request,

SOFTWARE
CONFIGURATION
MANAGEMENT

METODO SCRUM

- Analisi requisiti: esigenze e specifiche software
- Progettazione: creazione di una soluzione per il soddisfacimento requisiti
- Sviluppo: implementazione e scrittura del codice
- Testing: test delle parti principali del codice





REQUISITI

I requisiti sono stati elicitati tramite intervista diretta con un possibile cliente, un operatore in campo ospedaliero, e attraverso l'analisi delle funzionalità del software ospedaliero attualmente utilizzato in alcune strutture del territorio.

Tutti i dettagli relativi ai requisiti sono stati dettagliatamente descritti nel documento denominato "Documentazione del progetto.pdf".

Variante Stile MVC(Model-View-Controller)

La prospettiva iniziale consisteva nello strutturare il progetto utilizzando un'architettura a tre livelli: database, logico, interfaccia; in seguito il team ha ritenuto più opportuno optare per un altro stile architettonico: una variante dello stile Model-View-Controller.

In questa architettura il ruolo di Controller viene eseguito dal pacchetto "logico", che si appoggia a dati e funzionalità offerti dal pacchetto di gestione del database.


Il controller accetta le chiamate implicite provenienti dal livello View, identificato dal pacchetto "GUI", e attua dunque le necessarie manipolazioni sui dati utilizzando metodi JOOQ e/o i metodi forniti dalle classi del gestore del database (inserimento, rimozione e modifica).

I dati raccolti dal logico sono dunque caricati sul Model, rappresentato per l'appunto dal pacchetto "model", che permette la loro corretta visualizzazione sull'interfaccia utente.

La chiamata di aggiornamento della View con i dati del Modello è eseguita dal Controller.

Nel corso dell'implementazione del codice, si è scelto di applicare il pattern singleton per la creazione progetto database, assicurando che la classe possa avere una sola istanza al suo interno. Questo risultato è ottenuto mediante l'uso di un'unica istanza privata, un costruttore privato e, infine, un metodo pubblico che consente di richiamare l'istanza quando necessario.

// da definire



L'implementazione è completa, il sistema è completamente funzionante ed è eseguibile tramite il file main.java, presente nel progetto logica, all'interno della cartella "eseguibile"

Abbiamo utilizzato Eclipse sia per la creazione del database che per la realizzazione del codice in linguaggio java, inoltre è stato utilizzato il tool Window Builder per la creazione della GUI



IMPLEMENTAZIONE



DEMO

Procediamo con una breve dimostrazione del funzionamento dell'applicazione creata.

MODELLAZIONE



La presentazione della modellazione è avvenuta attraverso l'impiego dei seguenti diagrammi UML:

- Casi d'uso
- Attività
- Classe
- Sequenza
- Macchina a stati(Evidenziando gli stati del paziente)
 - Componenti
 - Diagramma ER
 - Package



TESTING

Per la creazione dei test inizialmente abbiamo utilizzato i test Junit automatico di Eclipse e successivamente implementato manualmente i vari test

//da definire

Quali risultati avete ottenuto con l'attività di testing