FATTURIFY

Gabriele Merli (1081373)
Lorenzo Colombo (1081134)
Carlo Alberto Poggiu (1079843)

Progetto ingegneria del software

FATTURIFY

11:11PM

Obiettivo

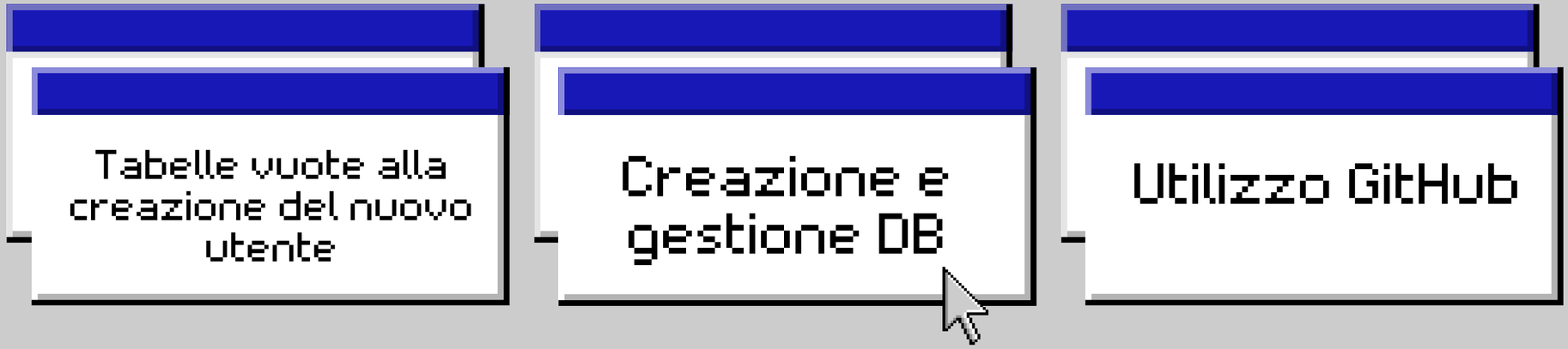
Fatturify nasce dall’esigenza di poter gestire l'organizzazione e la gestione dei cantieri. Questo strumento versatile consente una gestione efficiente del personale, del materiale e delle attrezzature, garantendo il massimo controllo e la massima efficienza in ogni fase del progetto.

Back to Agenda Page

Difficoltà Incontrate

Team: Il team si dimostra ben coordinato fin dall'inizio riuscendo a suddividere le mansioni sulla base degli accordi stabiliti precedentemente allo sviluppo dell'app

Sviluppo: Durante lo sviluppo dell'app sono state incontrate varie difficolta:



Risolto in fase di testing

Analisi con DB Browser

Abitudine

Programmazione e Modellazione

Per la creazione del programma è stato usato come linguaggio di programmazione Java e JUnit

Software Utilizzati

Start

Eclipse

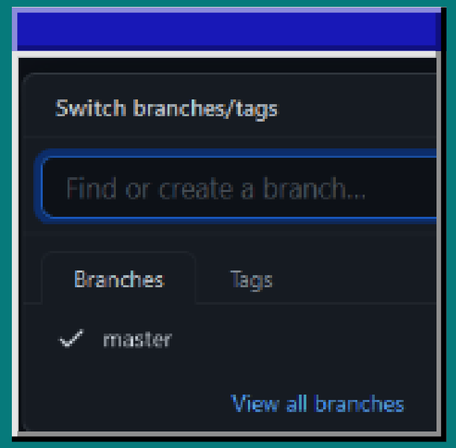
GitHub

Canva

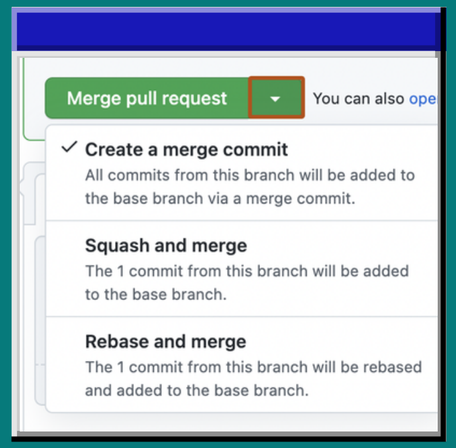
Star UML

Software Configuration Managment

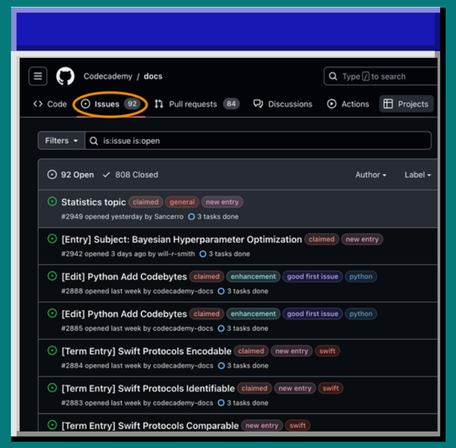
Abbiamo utilizzato github per coordinare il lavoro e mantenerlo monitorato utilizzando anche :



Branch



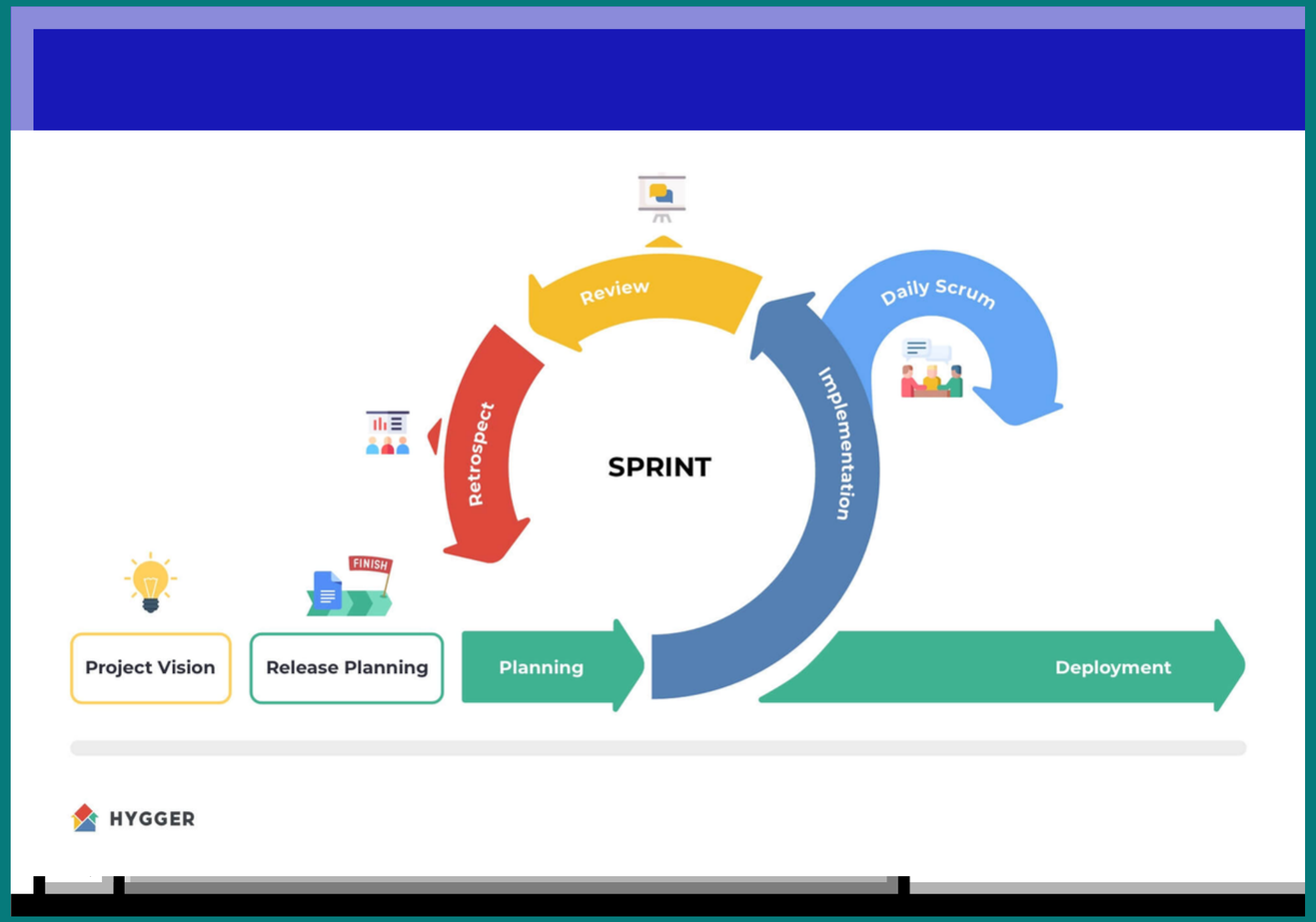
Pull



Issue per i vari errori/bug

Back to Agenda Page

Ciclo di vita del software



- Abbiamo programmato vari sprint nel corso del progetto accordati via chat
- Ogni fine sprint si testa la funzionalità delle modifiche apportate al progetto

Back to Agenda Page

Requisiti

- Per accedere all'applicazione è necessario il log-in del personale

-Creazione di Cantieri con inserimento Materiali/Personale e possibilità di aggiungere note testuali

-Gestione dell'Inventario con suddivisione ordinata di prodotti all'interno delle rispettive categorie

-Gestione del Personale con possibilità di aggiungere/modificare/eliminare un dipendente

Back to Agenda Page

Architettura

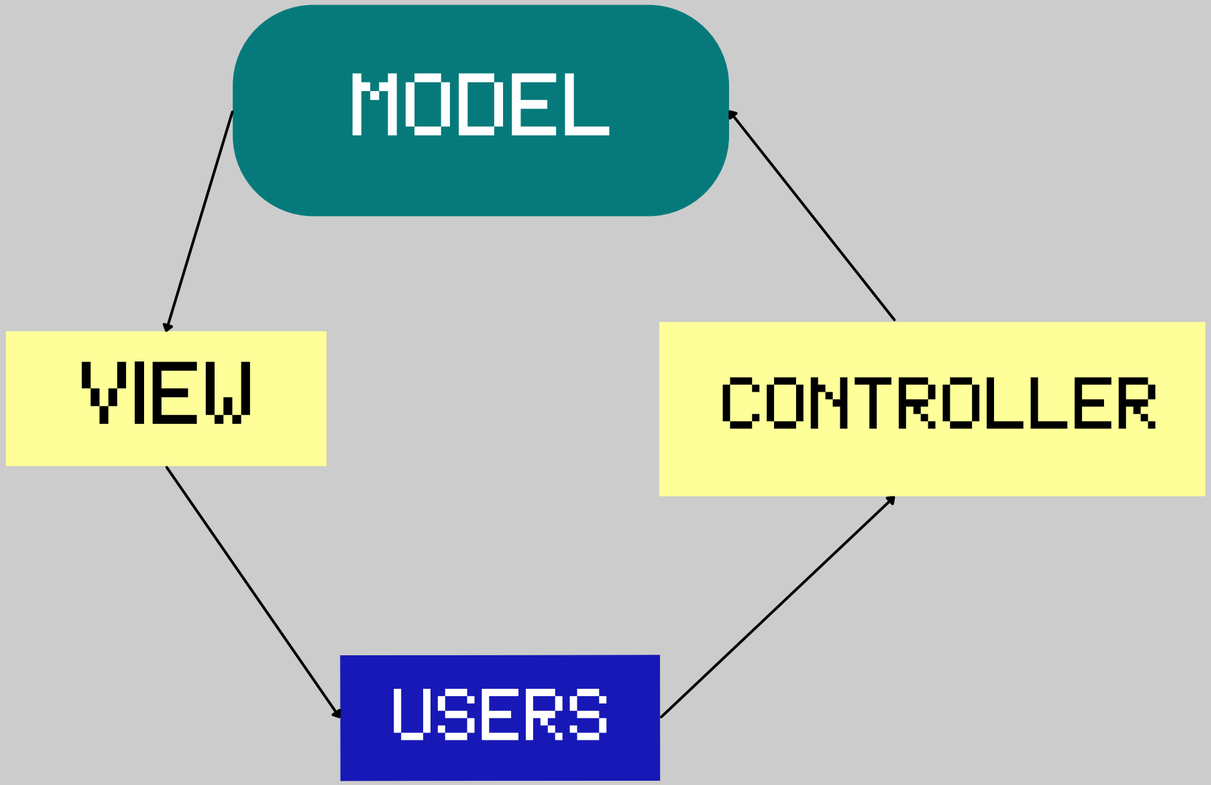
Per la realizzazione della nostra idea abbiamo deciso un architettura a micro-servizi per i seguenti motivi

- Flessibilità: Facile aggiungere o modificare funzionalità.

- Scalabilità: I servizi possono essere scalati indipendentemente a seconda delle necessità.

- Manutenzione: Aggiornamenti più semplici e minori rischi di interruzioni.

Design Pattern



MVC (Model-View-Controller) Pattern:

Separa l'applicazione in tre componenti principali:

il modello (dati),

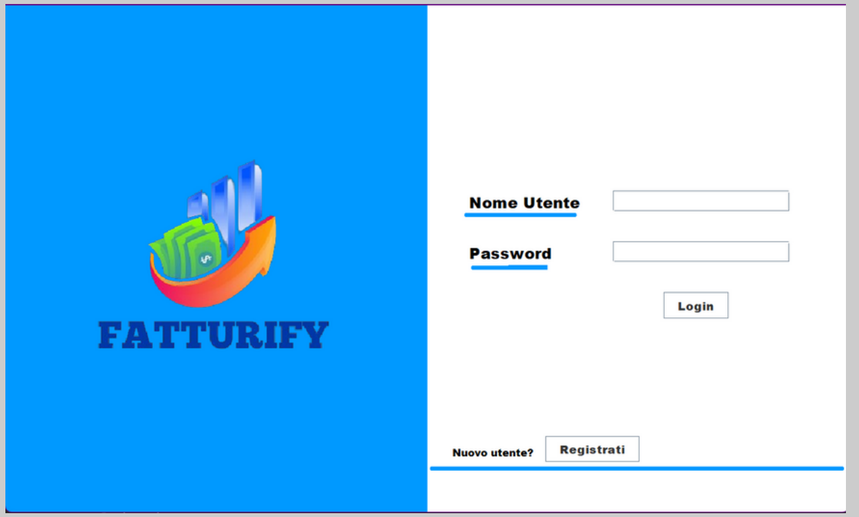
la vista (interfaccia utente)

e il controller (logica di business), facilitando così la gestione separata del codice in un'architettura a microservizi.

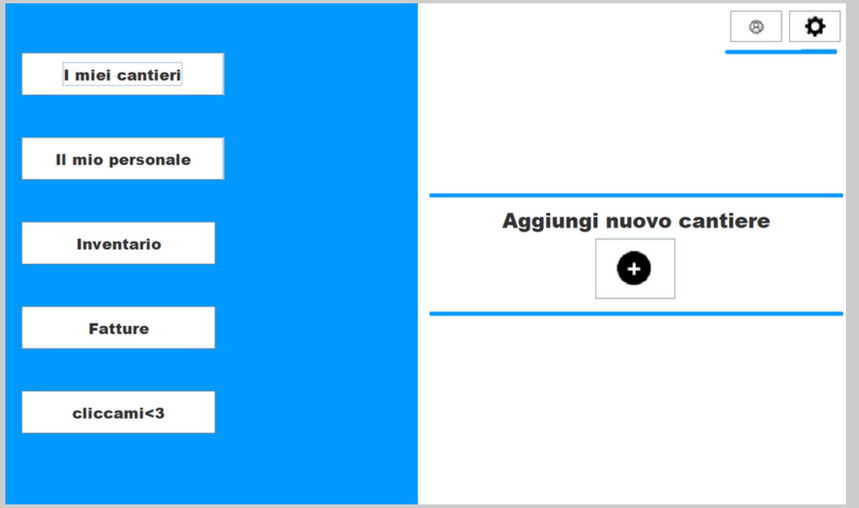
Implementazione

- Login Utente
- Autenticazione
- Pagina Home
- Cantieri
- Fatture
- Inventario
- Dipendenti

Demo

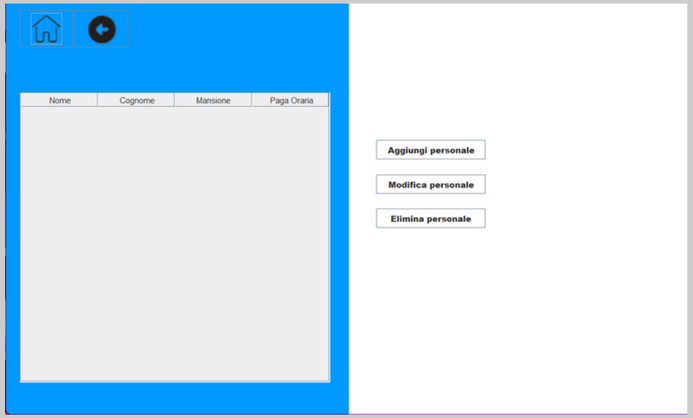


Login

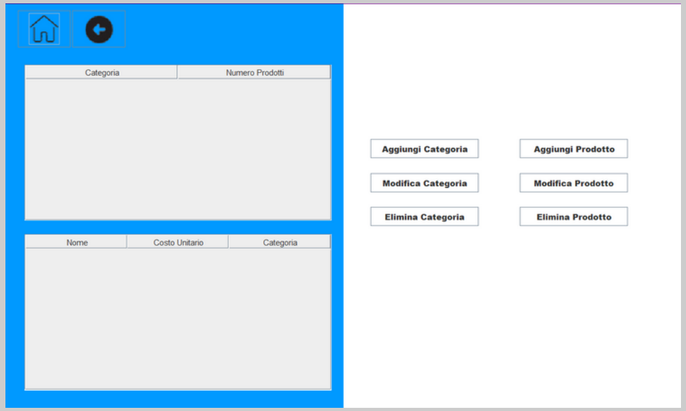


Home

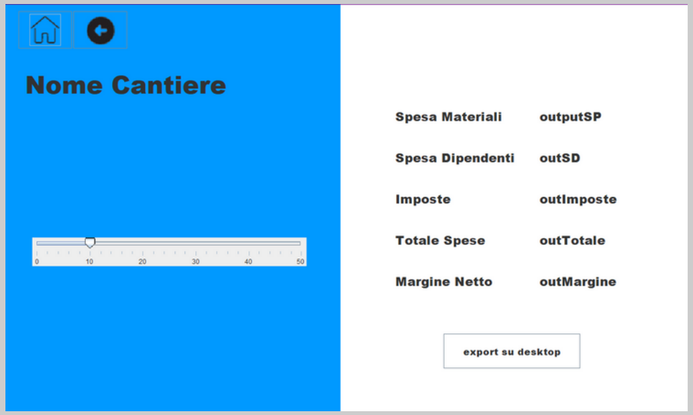
Demo



Personale



Inventario

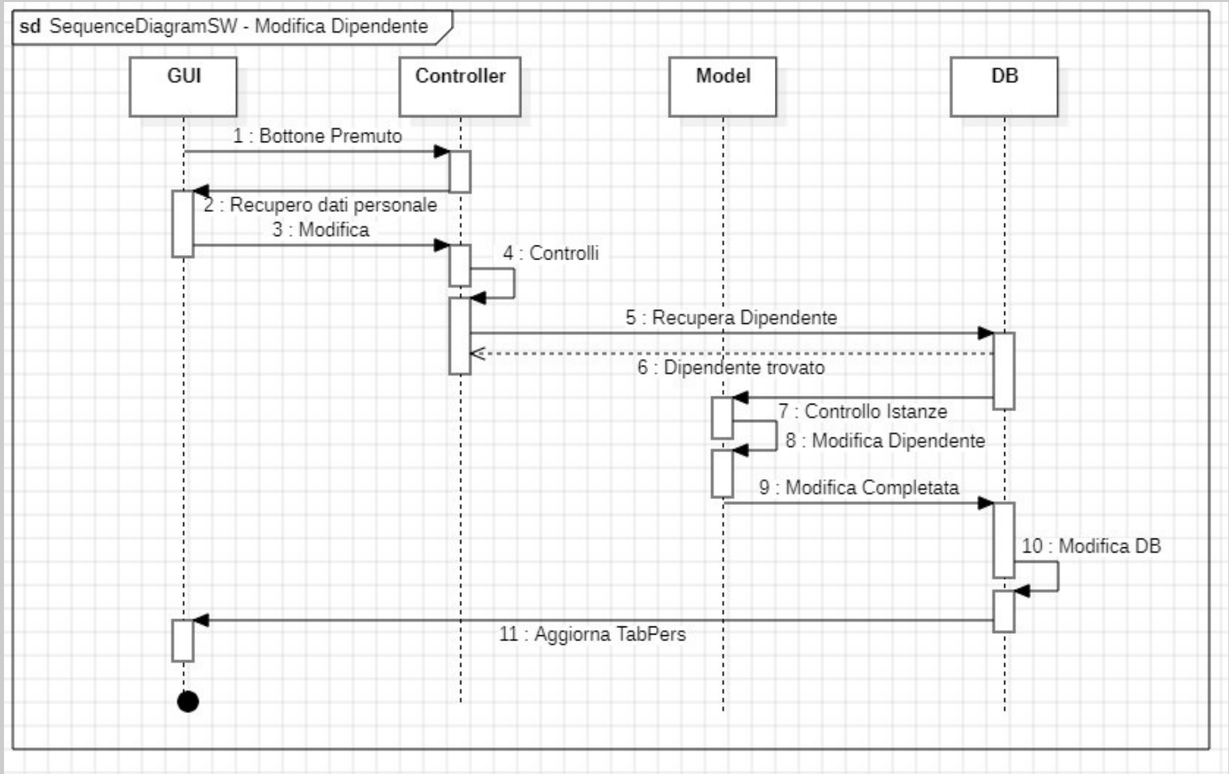


Fatture

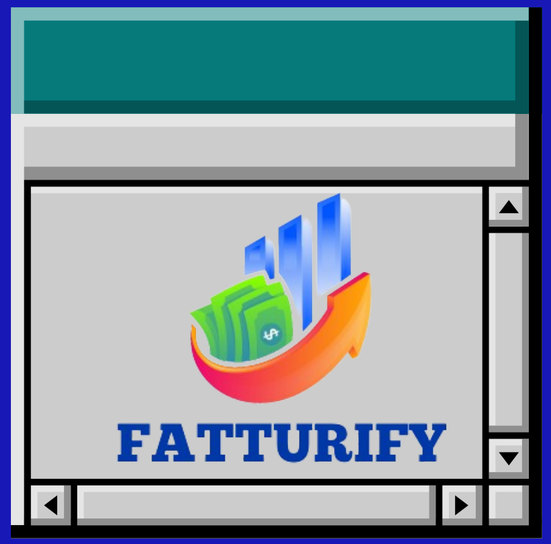
Modellazione

Per la modellazione abbiamo usato Star UML generando i seguenti diagrammi

- Casi d'uso
- Attività
- Classe
- Sequenza
- Macchina a stati
- Componenti



Testing

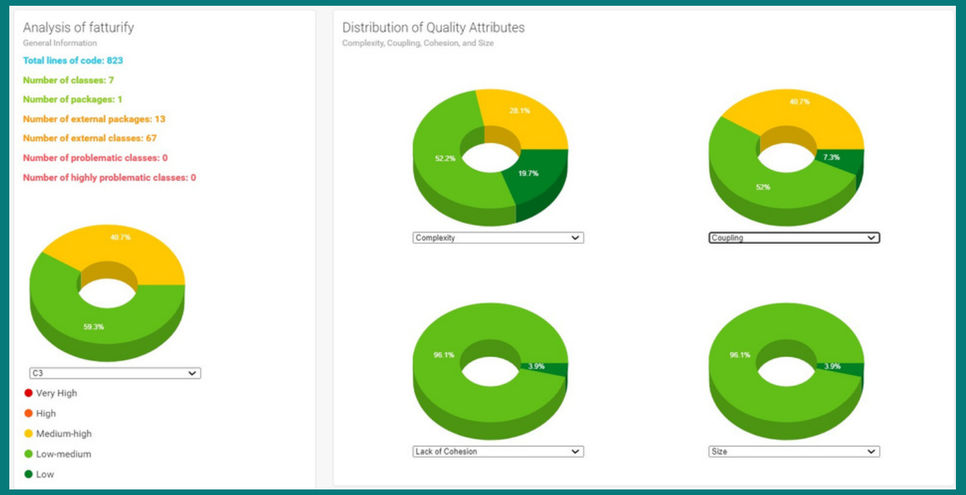


Durante lo sviluppo del software, abbiamo adottato un approccio incrementale nei test di funzionalità.

Ogni volta che una nuova funzionalità viene implementata, si procede immediatamente con i relativi test di funzionamento.

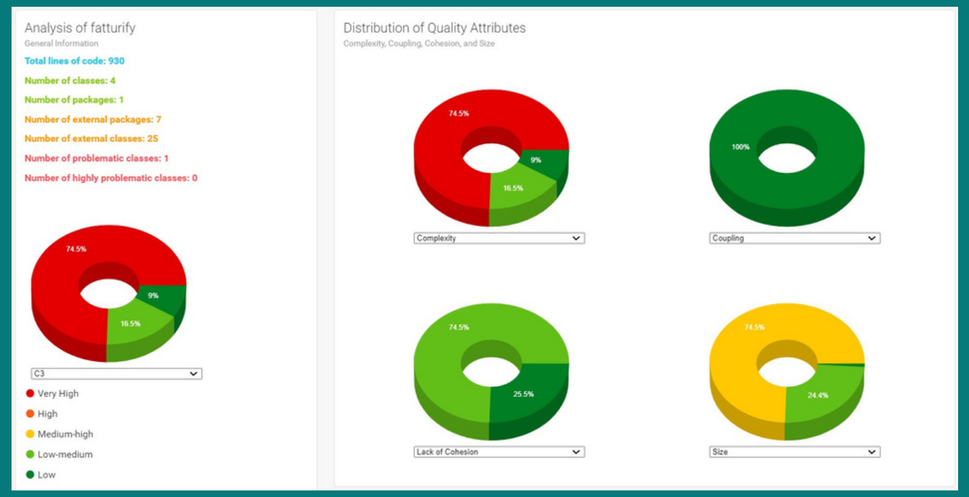
Questo processo ci ha permesso di identificare e correggere tempestivamente eventuali errori.

Code MR



Controller

Model



DataBase


Grazie Per
L'attenzione