Architettura e Design

Gabriele Chignoli

Giugno 2025

Contents

1 Architettura **3**
1.1 Architectural Views 3
1.1.1 Punti di vista del Modulo (Vista Statica) 3
1.1.2 Punti di vista dei Componenti e dei Connettori (Vista Dinamica) 4

2 Design **4**
2.1 Complessità 6

1 Architettura

Pantrymanager è un applicativo che prevede l'utilizzo di un'interfaccia grafica, per permettere all'utente di interagire con il sistema, e di un componente per conservare una moderata quantità di dati in modo consistente. Per questi motivi, lo stile architettonico **Model-View-Controller (MVC)** è stato scelto come modello più adeguato allo sviluppo dell'applicazione.

Attraverso il MVC, il sistema viene suddiviso in macro-componenti, i quali gestiscono ciascuno una certa funzionalità del software; tale suddivisione consente al sistema un buon grado di modularità, che aiuterà il team sia nelle fasi di sviluppo che in quelle di manutenzione.

Di seguito viene mostrata la suddivisione prevista dallo stile MVC.

- **Model** - si occupa di elaborare i dati salvati (produrre una dieta personalizzata) e gestire la comunicazione con il Database.
- **View** - mostra all'utente le funzionalità disponibili (pulsanti per aggiungere/modificare/rimuovere prodotti, creare una dieta personalizzata...), e fornisce l'output del Model all'utente (i parametri dei vari prodotti, i piatti disponibili...).
- **Controller** - gestisce gli input dell'utente, fornendo un feedback grafico, e comunica direttamente con il model.

1.1 Architectural Views

1.1.1 Punti di vista del Modulo (Vista Statica)

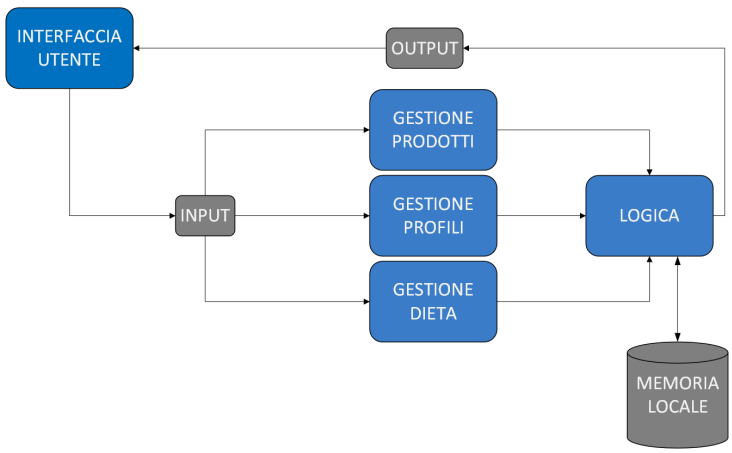

Figure 1: Punto di vista del modulo con relazione "Usa"

1.1.2 Punti di vista dei Componenti e dei Connettori (Vista Dinamica)

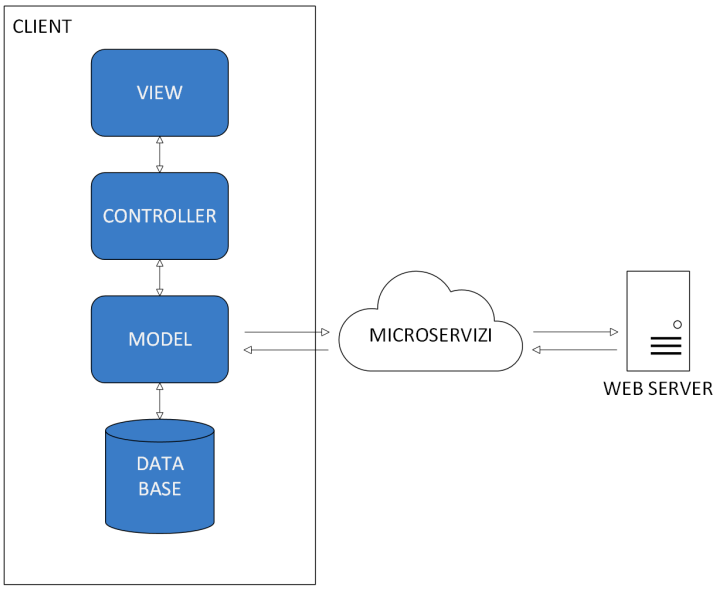

Figure 2: Punto di vista dei componenti e dei connettori

La struttura del client è quella che verrà effettivamente sviluppata e utilizzata come riferimento per lo sviluppo del software; i microservizi sono considerati un possibile ampliamento dell'architettura ma che non sarà implementata nella versione finale del software. Inoltre, le relazioni simboleggiate da due frecce distinte in direzione opposta indicano che la relazione è su richiesta, e non un continuo scambio di dati tra i componenti.

2 Design

Viene presentato di seguito il Diagramma delle Classi, volto a mostrare come si intende effettivamente implementare una delle parti costitutive del software. In particolare viene mostrato quali saranno le classi principali, i loro metodi e attributi e le loro relazioni (gerarchia) e interazioni (comunicazione).

Attraverso Papyrus è possibile generare il codice direttamente dal diagramma UML costruito, tuttavia son stati riscontrati errori (si pensa durante l'installazione) che non permettono il pieno funzionamento dello strumento. Per questo si è deciso di scrivere il codice manualmente utilizzando il diagramma come guida da seguire il più precisamente possibile; si tenterà di tenere aggiornate e coerenti entrambe le versioni.

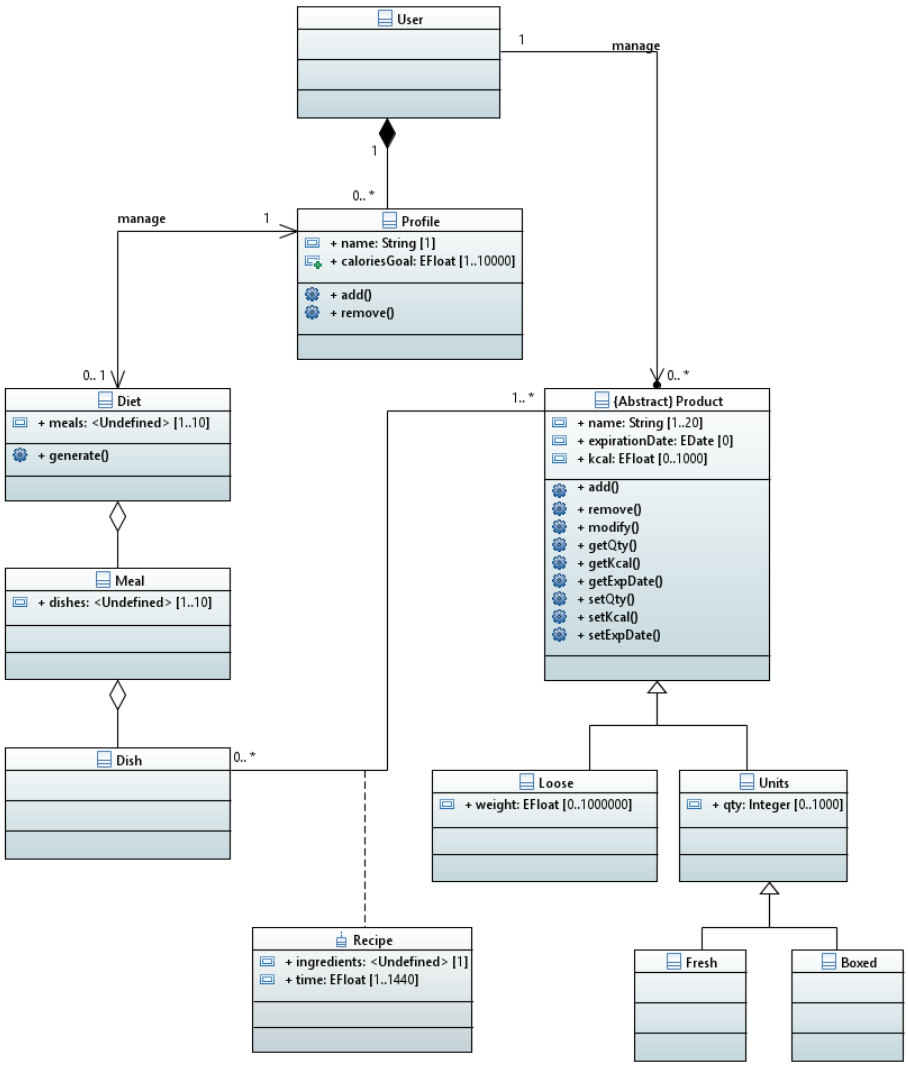

Figure 3: Diagramma delle Classi

Viene inoltre presentato il Diagramma delle Componenti al fine di mostrare le interazioni tra le macro-componenti del sistema e le interfacce che questi ultimi devono disporre e utilizzare per garantire il funzionamento dell'applicazione.

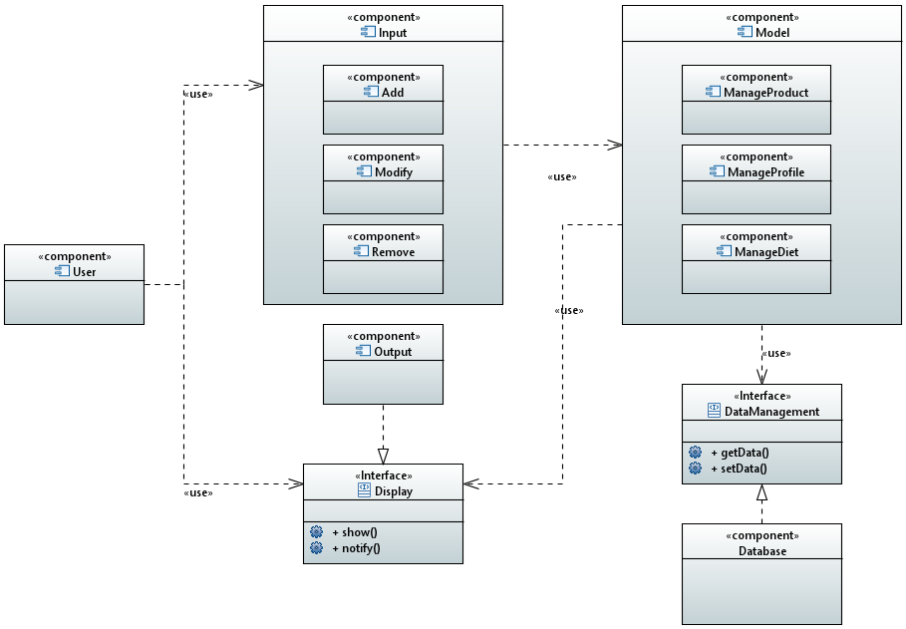

Figure 4: Component Diagram

2.1 Complessità