

Progetto di Ingegneria del Software T

BEAUTIFULWEIGHT

Candidati

Federico Baldassarre - 0000691911

Gabriele Corni - 0000691654

Federico Venturini – 0000693086

Descrizione del problema

L'applicativo *BeautifulWeight* si occupa di proporre e gestire diete ad hoc per raggiungere la forma fisica desiderata. Il sistema segue l'utente nella sua vita quotidiana proponendogli soluzioni alimentari personalizzate e personalizzabili, utilizzando algoritmi di ultima generazione e in continuo aggiornamento. Il programma supporta la multiutenza e si adatta alle esigenze di tutti gli utilizzatori, consentendo loro di scegliere piatti alternativi e indicare le proprie preferenze alimentari o allergie.

BeautifulWeight: siate il vostro io migliore.

Documento dei requisiti

Si vuole realizzare una piattaforma software per distribuire servizi dietetici. L'azienda intende distribuire l'applicazione in più versioni iniziando da una versione gratuita che permette l'utilizzo da parte di un solo utente e impone altre restrizioni sull'uso, e una versione a pagamento che permette di aggiungere più utenti e sfruttare tutti i servizi.

Alla creazione di ogni profilo utente è richiesta la compilazione di una scheda anagrafica e fisiologica contenente:

- Nome
- Cognome
- Data di nascita
- Peso
- Altezza
- Sesso
- Ore di lavoro settimanale
- Carico lavorativo (scala 1-5 tra lavoro di scrivania e lavoro manuale intenso)
- Ore di attività sportiva settimanale
- Carico sportivo (scala 1-5 di intensità agonistica)

Nella versione *premium* è inoltre possibile specificare:

- *Obiettivo della dieta*: a scelta tra mantenimento peso, aumento massa muscolare, perdita di peso, definizione muscolare (nella versione standard viene permesso di utilizzare unicamente l'obiettivo perdita di peso).
- *Preferenze alimentari*: in termini di ingredienti da escludere dal calcolo della dieta a seguito di gusti alimentari o intolleranze (nella versione standard sono considerati tutti gli ingredienti in archivio).

In base ai dati contenuti nel profilo utente il sistema calcola automaticamente il peso forma ideale in base a una formula. La formula attualmente utilizzata per il calcolo del peso forma è la *formula di Keyes*.

Essendo il software destinato ad un utilizzo privato o familiare non è necessario introdurre forme di privacy nella gestione dei vari profili; ogni utilizzatore deve quindi poter gestire un qualsiasi profilo inserito nel sistema in termini di visualizzazione, modifica, cancellazione e creazione.

Il software deve essere distribuito nella sua versione gratuita, con eventuale *upgrade* per sbloccare ulteriori funzioni effettuabile direttamente *in-app*. Ogni licenza è permanente. Non è possibile effettuare *downgrade*.

Il meccanismo di upgrade avviene tramite l'inserimento e la validazione di un codice segreto, fornito all'utente in maniera esterna all'applicativo (ad esempio per email a seguito di un pagamento, oppure sotto forma di coupon gratuito). Se la validazione del codice inserito ha successo, la nuova versione è sbloccata.

Per ogni profilo utente è possibile calcolare un menù settimanale basato su una certa tipologia di dieta. L'azienda dispone di diverse tipologie di dieta, compatibili o meno con i vari obiettivi. Si garantisce che per ogni obiettivo sia sempre disponibile almeno una dieta mirata al suo raggiungimento. Indipendentemente dalla versione, è sempre possibile scegliere la dieta desiderata fra quelle compatibili con l'obiettivo inserito.

In seguito è riportato l'elenco delle diete attualmente supportate dall'azienda, unitamente alle compatibilità con gli obiettivi.

	Mantenimento	Massa muscolare	Perdita di peso	Definizione
Dieta a zona	✓	✗	✓	✓
Dieta a punti	✓	✓	✓	✓
Dieta a kcal	✓	✓	✓	✓
Dieta Dukan	✗	✗	✓	✗
Dieta dissociata	✗	✗	✓	✗
Crono dieta	✗	✓	✓	✗
Dieta ipocalorica	✗	✗	✓	✓
Dieta vegana	✓	✗	✓	✓

Gli algoritmi per il calcolo delle diete sono proprietari e sviluppati internamente all'azienda utilizzando il framework .NET e distribuite sotto forma di librerie a caricamento dinamico. Al fine di permettere il *testing* del software l'azienda mette a disposizione alcuni algoritmi di prova. Si prevede che in futuro possano essere aggiunte nuove tipologie di dieta.

Nella versione standard sono forniti i servizi di *calcolo* di un nuovo menù settimanale (che eventualmente sovrascrive il precedente) e di *visualizzazione* di peso forma e dieta corrente; nella versione premium è inoltre possibile *visualizzare* il pasto successivo in base a ora e giorno della settimana e *sostituire un piatto* del menù settimanale con uno equivalente.

Ogni menù è basato su 7 giorni, ognuno dei quali prevede una serie di pasti a diversi orari, ogni pasto si compone di piatti in una certa quantità.

Un piatto è composto da un elenco di ingredienti e ad ogni piatto è associato un apporto nutrizionale (calorie, proteine, carboidrati, grassi) basato su 100gr di prodotto alimentare.

Un ingrediente è caratterizzato unicamente dal suo nome e non dai suoi valori nutrizionali. Esempi di ingredienti sono: lattuga, pasta, salsa di pomodoro. Non volendo realizzare un ricettario, non si indicano le quantità dei singoli ingredienti componenti un piatto.

Glossario

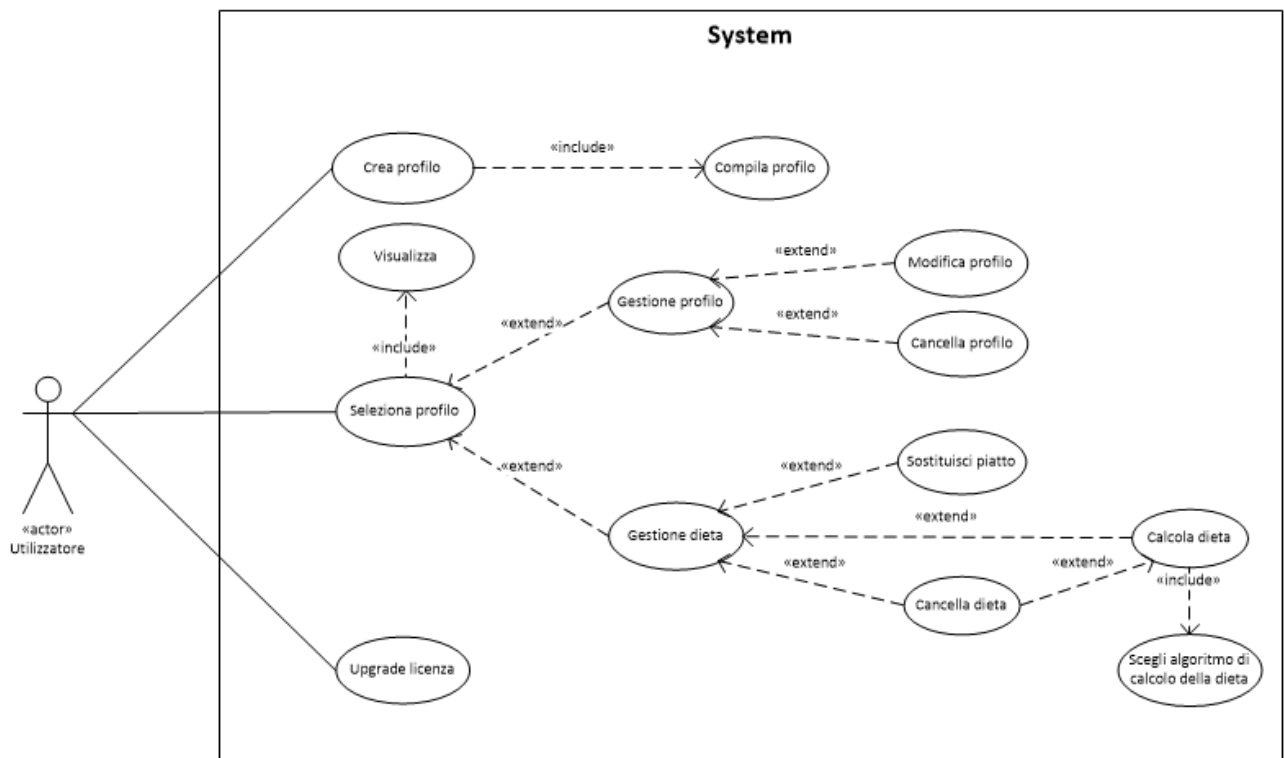
Termine	Significato e caratteristiche
Versione standard Versione gratuita	Versione del programma che impone le seguenti limitazioni: <ul style="list-style-type: none"> - Gestire al massimo un profilo utente. - Non è possibile modificare obiettivo e preferenze alimentari del profilo. - Non è possibile visualizzare il pasto successivo. - Non è possibile scegliere piatti alternativi.
Versione premium Versione a pagamento	Versione del programma senza le limitazioni della versione standard.
Utente Utilizzatore	Persona che utilizza l'applicativo e che può interagire con esso. Può fare le seguenti azioni: <ul style="list-style-type: none"> - Visualizzare, modificare, aggiungere e cancellare un profilo utente - Estendere la versione del prodotto
Profilo utente	Una descrizione completa delle caratteristiche rilevanti di un utilizzatore, contiene scheda anagrafica e fisiologica, obiettivo e preferenze alimentari.
Scheda anagrafica e fisiologica	Informazioni necessarie per il calcolo della dieta inserite dall'utente all'atto della creazione di un nuovo profilo. Comprende: <ul style="list-style-type: none"> - Nome - Cognome - Data di nascita - Peso (espresso in Kg) - Altezza (espresso in cm) - Sesso - Ore di lavoro settimanale - Carico lavorativo - Ore di attività sportiva settimanale - Carico sportivo
Obiettivi	Ciò che l'utente desidera ottenere tramite la dieta. Sono previsti gli obiettivi di: <ul style="list-style-type: none"> - Mantenimento peso - Dimagrimento - Aumento massa muscolare - Definizione muscolare
Preferenze Alimentari	Ingredienti da escludere dal calcolo della dieta.
Carico sportivo	Carico dell'attività sportiva indicato su una scala di valori da 1 a 5 dove 1 indica attività a bassa intensità e 5 indica l'estremo opposto. (esempio: Yoga – 1, Calcio – 3, Triathlon – 5)
Carico lavorativo	Carico lavorativo indicato su una scala di valori da 1 a 5 dove 1 indica lavoro di scrivania e 5 indica lavoro manuale intenso. (esempio: Impiegato – 1, Commesso – 3, Muratore - 5)
Gestione profilo utente	Operazione generica di accesso a un profilo utente al fine di visualizzarlo, modificarlo, cancellarlo oppure crearne uno nuovo.
Compilazione profilo utente	Operazione di riempimento di tutti i campi di un profilo utente.

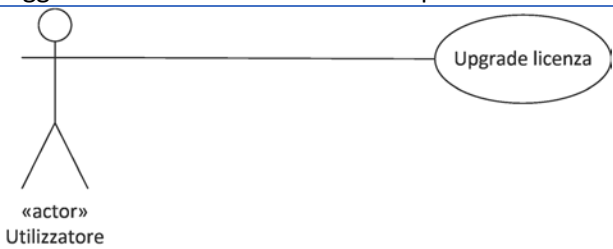
Visualizzazione profilo utente	Operazione di visualizzazione di un profilo utente. Contestualmente viene visualizzato il peso forma calcolato utilizzando la formula del peso forma e la dieta eventualmente ad esso associata.
Modifica profilo utente	Operazione di modifica dei dati del profilo utente o di ricalcolo della dieta.
Creazione profilo utente	Operazione di creazione di un nuovo profilo utente. Durante la creazione è necessario effettuare la compilazione del nuovo profilo.
Cancellazione profilo utente	Operazione di rimozione di un profilo utente dal sistema.
Estensione della versione Upgrade	Operazione di upgrade della versione del software. Prevede l'inserimento di un codice di attivazione da parte dell'utente in seguito alla validazione del quale vengono sbloccati i servizi aggiuntivi. Il codice viene comunicato all'utente esternamente al sistema. Le licenze sono permanenti.
Feature Funzionalità	Funzionalità offerta dal sistema che può essere abilitata o meno a seconda della versione. Le funzionalità finora dichiarate sono: aggiunta di più di un utente, cambio dell'obiettivo, cambio delle preferenze alimentari, sostituzione di un piatto, visualizzazione pasto successivo.
Dieta Menù settimanale	Programma settimanale dei pasti. Ogni giorno prevede una serie di pasti diversi a diversi orari.
Visualizzazione della dieta corrente	Stampa a video del menù settimanale.
Scelta di una tipologia di dieta	Scelta dell'algoritmo di calcolo di una dieta.
Calcolo di una dieta	Operazione di formulazione di un menù settimanale per un utilizzatore in base ai dati inseriti in un profilo utente tramite un algoritmo di calcolo della dieta.
Algoritmo di calcolo di una dieta Tipologia di dieta	Strumento proprietario fornito dall'azienda atto al calcolo di un menù settimanale sulla base di un profilo utente dato. Fornisce inoltre la possibilità di trovare un piatto equivalente ad uno dato in base alla logica specifica del tipo di dieta.
Algoritmo del peso forma Formula del peso forma	Strumento di calcolo del peso forma di un utente a partire dal suo profilo.
Visualizzazione del pasto successivo	Servizio di visualizzazione del pasto successivo previsto dal menù settimanale.
Scelta di un piatto alternativo	Operazione di sostituzione di un piatto all'interno di un pasto con uno ad esso equivalente. L'equivalenza dipende dal tipo di dieta.
Ingrediente	Elemento costitutivo base di un piatto, caratterizzato unicamente dal suo nome.
Piatto	Composto da un elenco di ingredienti e associato ad un apporto nutrizionale basato su 100gr di prodotto alimentare.
Valori Nutrizionali	Quantità di Calorie, proteine, carboidrati, grassi su cui basarsi per il calcolo della dieta.
Pasto	Insieme di uno o più piatti, ognuno con un quantitativo in grammi e da consumare all'orario specificato.

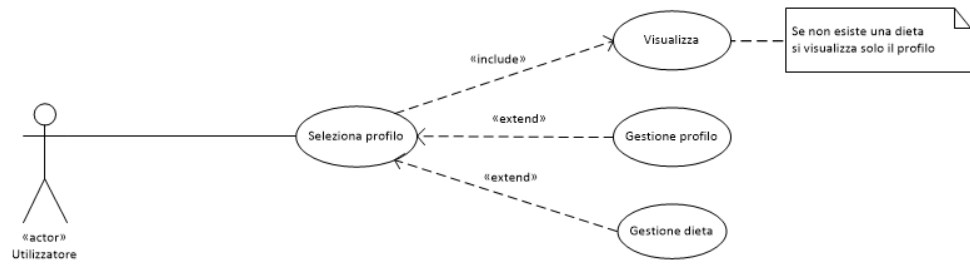
Casi d'uso e scenari

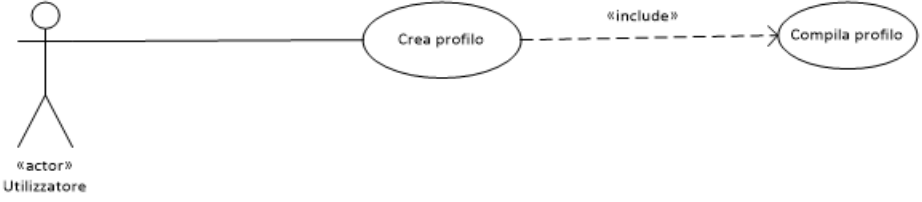
Note comuni

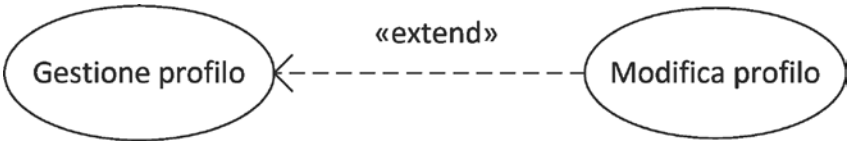
- Per ogni operazione soggetta a restrizioni di licenza è necessario verificare preventivamente la versione del prodotto
- Ogni operazione può essere annullata dall'utente
- Tutte le operazioni critiche sono soggette a richiesta di conferma

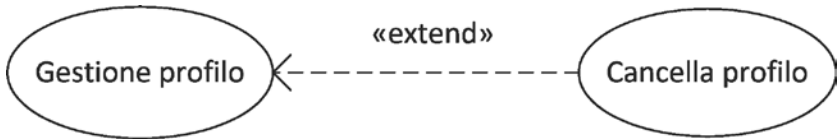


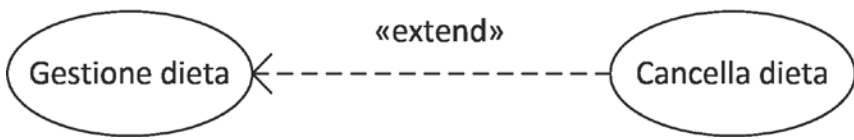
Titolo	Upgrade della versione di sistema
Descrizione	Aggiornamento ad una versione superiore del software
Relazioni	
Attori	Utente
Precondizioni	Si dispone di una versione che non sia la più elevata disponibile in commercio
Postcondizioni	Le funzionalità aggiuntive della versione ottenuta sono accessibili
Scenario Principale	<ol style="list-style-type: none"> 1. L'utente seleziona l'operazione di upgrade 2. L'utente seleziona la versione di interesse (superiore a quella posseduta) 3. L'utente inserisce il codice di aggiornamento per la versione indicata 4. Il sistema verifica la validità del codice 5. Il sistema sblocca le funzionalità aggiuntive della nuova versione
Scenari alternativi	<ol style="list-style-type: none"> 3. <ol style="list-style-type: none"> a. Il codice inserito non è valido <ol style="list-style-type: none"> i. Il sistema notifica l'errore e torna al passo 2
Requisiti non funzionali	
Punti Aperti	<p>Qual è il formato del codice?</p> <p>Come avviene la validazione?</p>

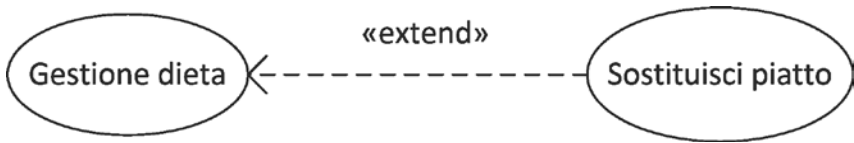
Titolo	Selezione un profilo utente
Descrizione	L'utilizzatore seleziona il profilo utente di interesse e contestualmente visualizza le informazioni associate
Relazioni	
Attori	Utente
Precondizioni	E' stato creato almeno un profilo utente
Postcondizioni	
Scenario Principale	<ol style="list-style-type: none"> 1. L'utente seleziona il profilo utente da visualizzare 2. Il sistema mostra a video i dati del profilo utente selezionato 3. Se esiste una dieta associata al profilo selezionato, il sistema la mostra a video insieme al pasto successivo
Scenari alternativi	
Requisiti non funzionali	
Punti Aperti	

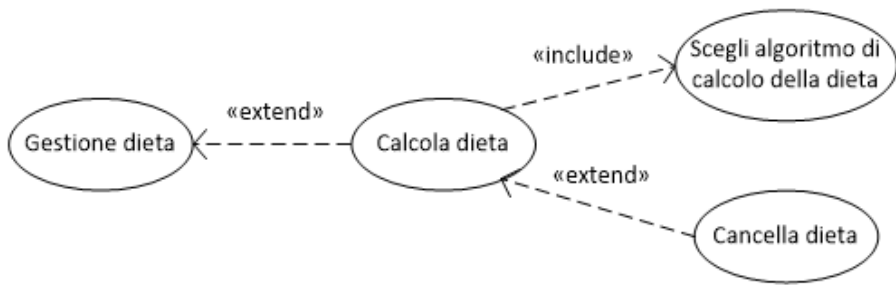
Titolo	Crea un profilo utente
Descrizione	Creazione e compilazione di un nuovo profilo utente con il quale poter utilizzare il sistema
Relazioni	 <pre> graph LR Actor[«actor» Utilizzatore] --- UC1((Crea profilo)) UC1 -.-> «include» UC2((Compila profilo)) </pre>
Attori	Utente
Precondizioni	La versione del programma permette l'aggiunta di un nuovo profilo utente
Postcondizioni	Un nuovo profilo utente è stato creato e compilato in tutti i suoi campi ed è disponibile nella selezione utenti
Scenario Principale	<ol style="list-style-type: none"> 1. L'utente avvia la creazione di un nuovo profilo 2. L'utente compila tutti i campi che gli è permesso compilare in base alla versione 3. L'utente conferma i dati inseriti
Scenari alternativi	<ol style="list-style-type: none"> 3. <ol style="list-style-type: none"> a. Se esiste un profilo utente con lo stesso nome <ol style="list-style-type: none"> i. Il sistema notifica l'errore e torna al passo 2
Requisiti non funzionali	
Punti Aperti	<p>Qual è l'unità di misura del peso?</p> <p>Qual è l'unità di misura dell'altezza?</p> <p>Da che cosa è identificato univocamente un profilo utente? Che cos'è un profilo utente con lo stesso nome? Omonimi?</p>

Titolo	Modifica un profilo
Descrizione	L'utente modifica le informazioni associate ad un profilo utente
Relazioni	 <pre> graph LR UC1((Gestione profilo)) -.-> «extend» UC2((Modifica profilo)) </pre>
Attori	Utente
Precondizioni	E' stato selezionato un profilo utente
Postcondizioni	I dati del profilo utente selezionato sono modificati
Scenario Principale	<ol style="list-style-type: none"> 1. L'utente seleziona l'operazione di modifica 2. L'utente modifica le informazioni desiderate associate al profilo utente (con ricalcolo del peso forma) 3. L'utente conferma l'operazione
Scenari alternativi	
Requisiti non funzionali	
Punti Aperti	Vedi Creazione profilo utente

Titolo	Cancella un profilo
Descrizione	L'utente modifica le informazioni associate ad un profilo utente
Relazioni	 <pre> graph LR A((Gestione profilo)) B((Cancella profilo)) B -.-> «extend» A </pre>
Attori	Utente
Precondizioni	E' stato selezionato un profilo utente
Postcondizioni	Il profilo utente viene rimosso dal sistema
Scenario Principale	<ol style="list-style-type: none"> 1. L'utente seleziona l'operazione di cancellazione 2. L'utente conferma l'operazione di cancellazione del profilo utente selezionato
Scenari alternativi	
Requisiti non funzionali	
Punti Aperti	

Titolo	Cancella la dieta corrente
Descrizione	L'utente cancella la dieta correntemente associata ad un profilo utente
Relazioni	 <pre> graph LR A((Gestione dieta)) B((Cancella dieta)) B -.-> «extend» A </pre>
Attori	Utente
Precondizioni	E' stato selezionato un profilo utente che ha una dieta associata
Postcondizioni	Il profilo utente selezionato non ha più una dieta associata
Scenario Principale	<ol style="list-style-type: none"> 1. L'utente seleziona l'operazione di cancellazione 2. L'utente conferma l'operazione di cancellazione dalla dieta associata al profilo utente selezionato
Scenari alternativi	
Requisiti non funzionali	
Punti Aperti	

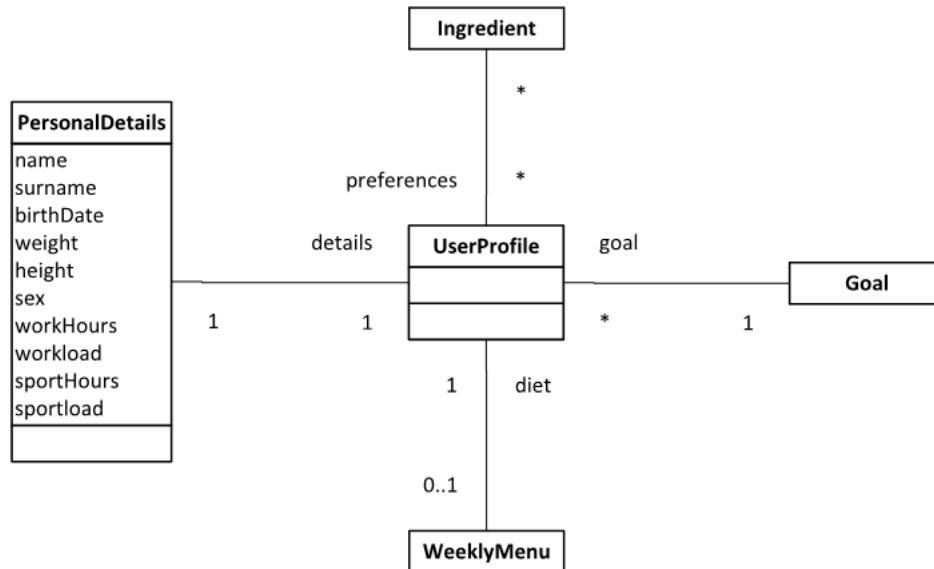
Titolo	Sostituisci un piatto
Descrizione	L'utente richiede la sostituzione di un piatto nel menù settimanale con un piatto alternativo equivalente
Relazioni	 <pre> graph LR A((Gestione dieta)) -.-> «extend» B((Sostituisci piatto)) </pre>
Attori	Utente
Precondizioni	E' stato selezionato un profilo utente che ha una dieta associata
Postcondizioni	Un piatto della dieta associata al profilo utente selezionato è stato sostituito con un piatto equivalente
Scenario Principale	<ol style="list-style-type: none"> 1. L'utente seleziona un piatto su cui effettuare la sostituzione 2. L'utente sceglie tra un elenco di piatti alternativi proposti dal sistema
Scenari alternativi	<ol style="list-style-type: none"> 2. <ol style="list-style-type: none"> a. Non esiste nessun piatto equivalente a quello selezionato i. Il sistema lascia il menù settimanale invariato
Requisiti non funzionali	
Punti Aperti	

Titolo	Calcola dieta
Descrizione	L'utente richiede il calcolo di un nuovo menù settimanale associato al profilo utente
Relazioni	 <pre> graph LR A((Gestione dieta)) -.-> «extend» C((Calcola dieta)) B((Cancella dieta)) -.-> «extend» C C -.-> «include» D((Scegli algoritmo di calcolo della dieta)) </pre>
Attori	Utente
Precondizioni	E' stato selezionato un profilo utente
Postcondizioni	Il profilo utente ha una nuova dieta associata
Scenario Principale	<ol style="list-style-type: none"> 1. L'utente seleziona l'operazione di calcolo di una dieta 2. Se esiste già una dieta associata al profilo, il sistema chiede se si vuole effettuare la sostituzione della dieta <ol style="list-style-type: none"> a. Se la risposta è positiva il sistema cancella la dieta precedente e procede b. Altrimenti il caso d'uso termina 3. L'utente seleziona l'algoritmo di calcolo della dieta in accordo coi dati del profilo utente 4. Il sistema calcola una nuova dieta e la associa al profilo utente
Scenari alternativi	<ol style="list-style-type: none"> 4. <ol style="list-style-type: none"> a. Il sistema non riesce a calcolare una nuova dieta basandosi sul profilo utente i. Il sistema notifica l'errore e termina il caso d'uso
Requisiti non funzionali	
Punti Aperti	Nella visualizzazione della nuova dieta, si vuole permettere di sostituire un piatto?

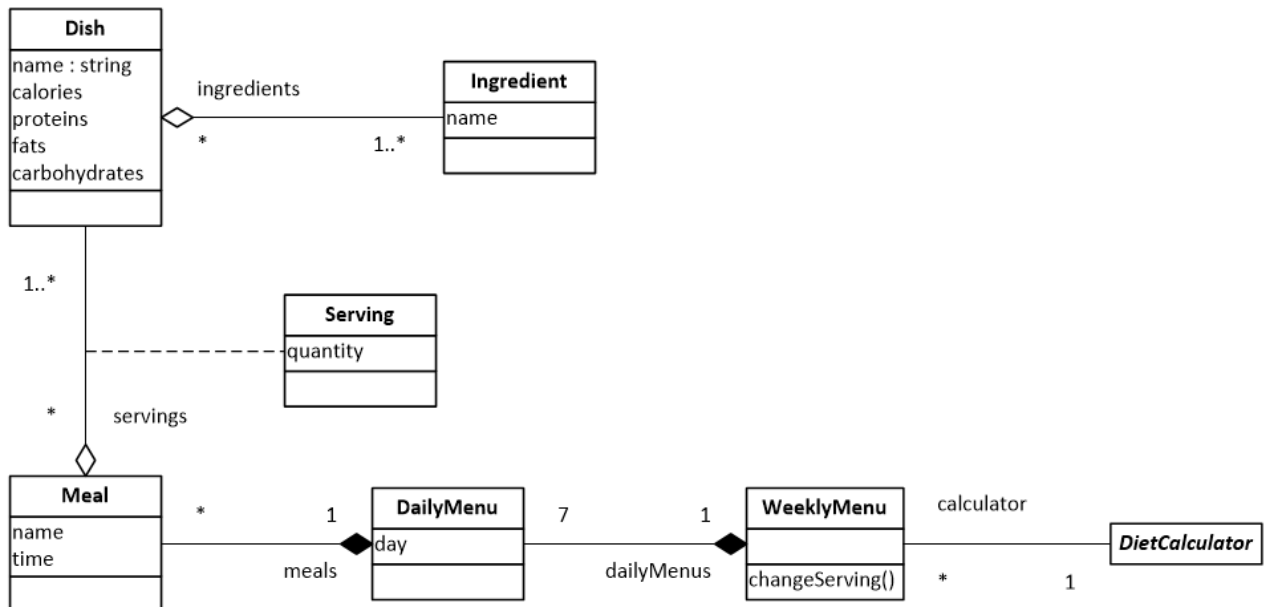
Modello statico

Si noti che le visibilità non sono state inserite poiché si utilizzano le regole dell'analisi di default (metodi pubblici, campi privati).

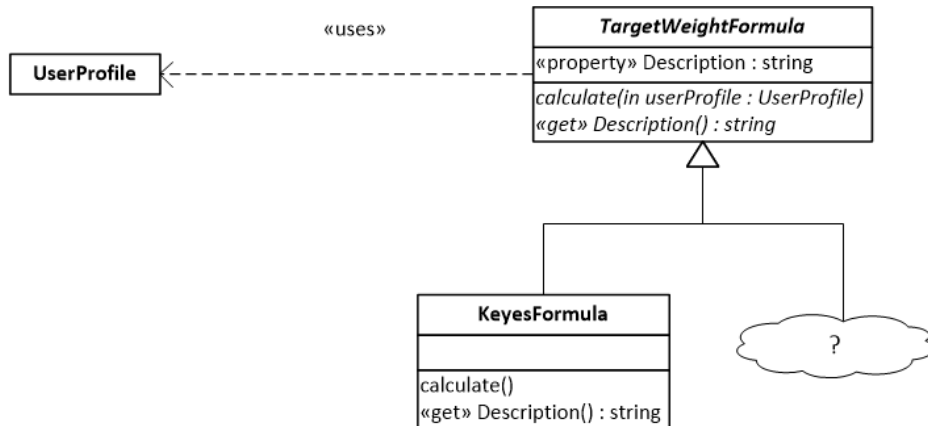
User Profile:



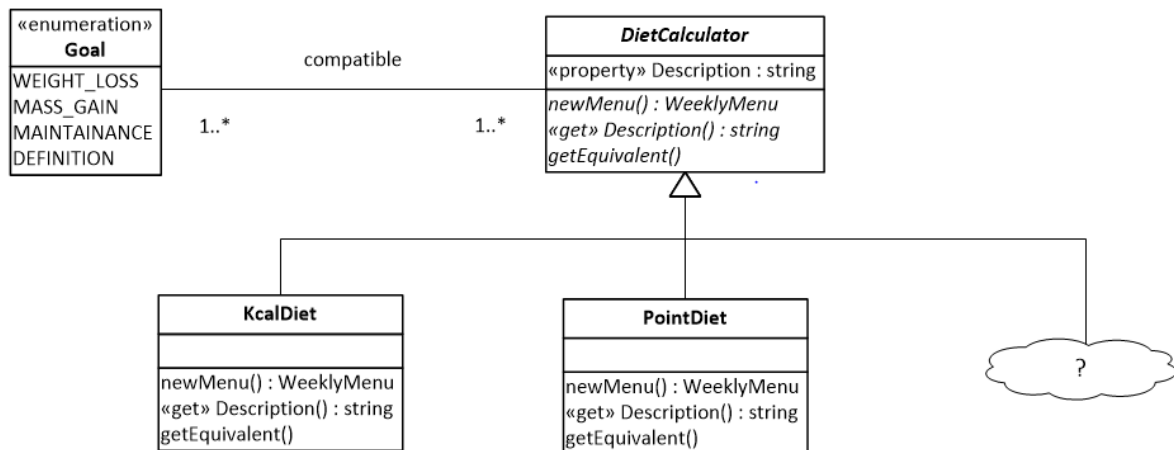
Weekly Menu:



TargetWeightFormula:



Diet Calculator:



Version and code validating:

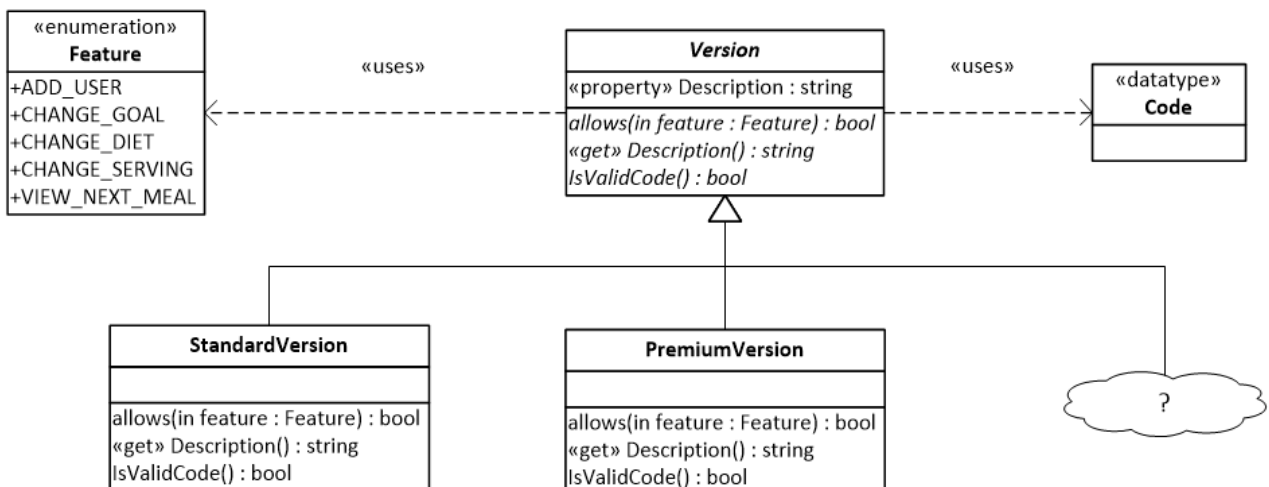
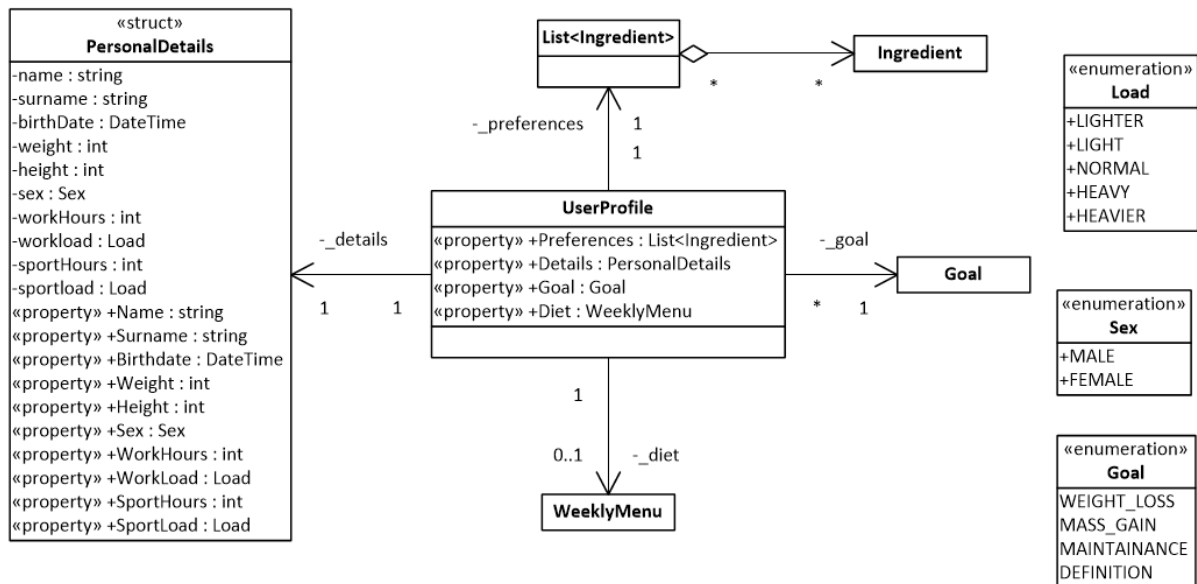
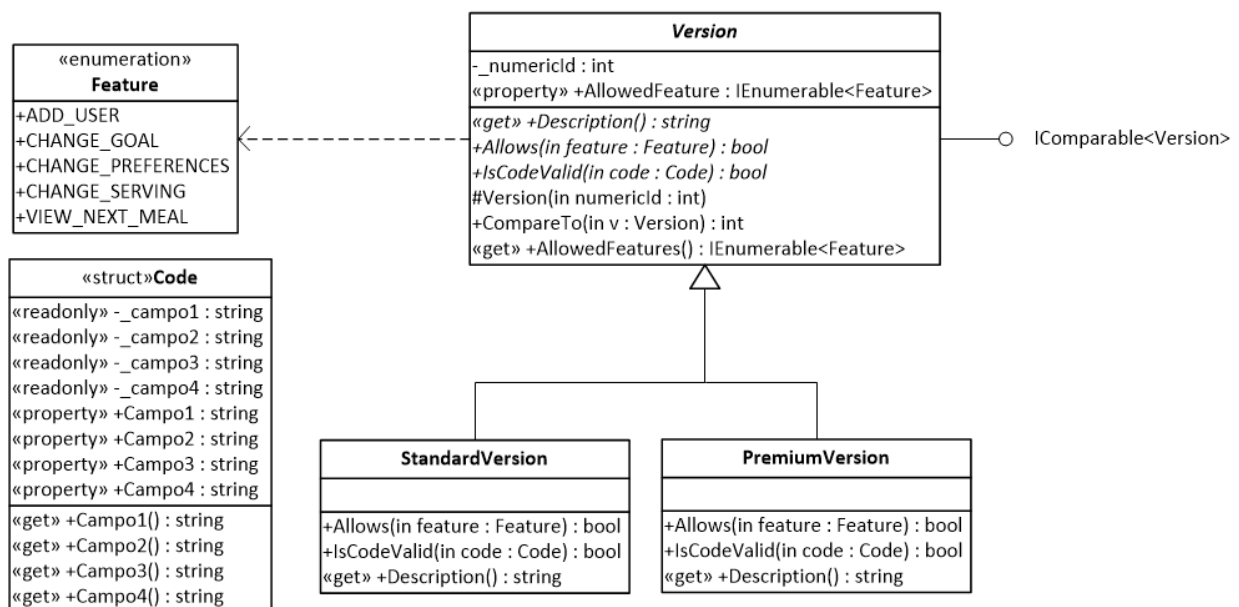


Diagramma delle classi di progettazione

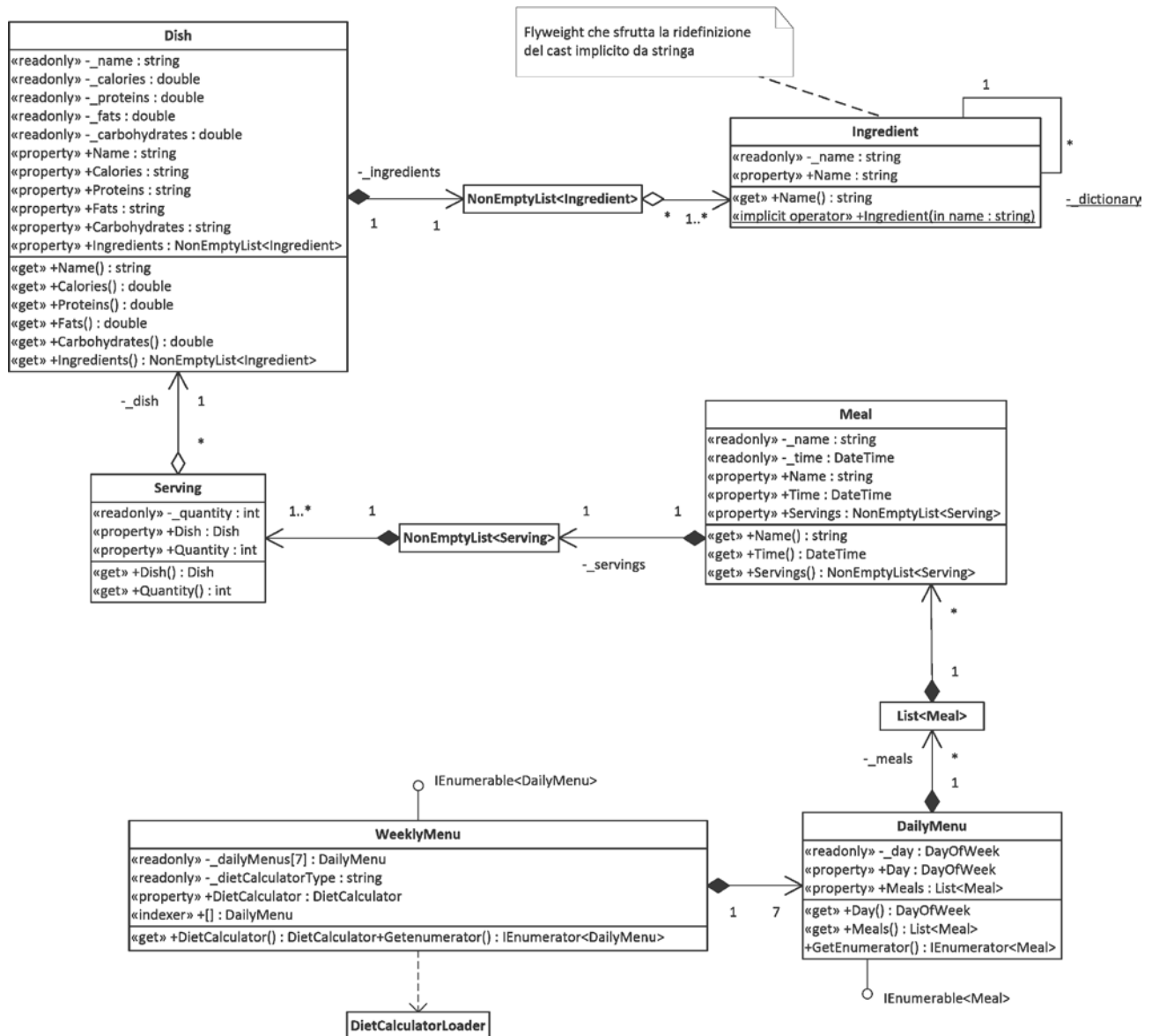
UserProfile ed Enumerativi:



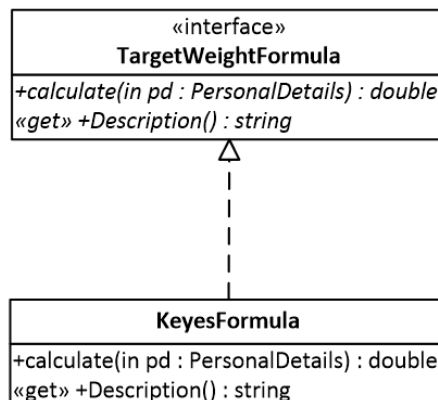
Versione:



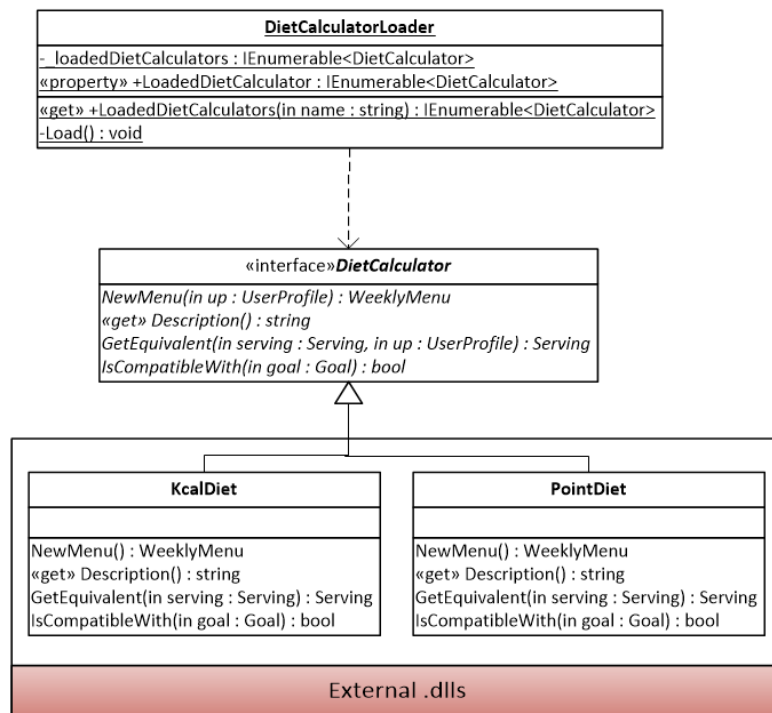
Ingredienti, Piatti, Menu Giornalieri e Settimanali:



Calcolo del peso forma:

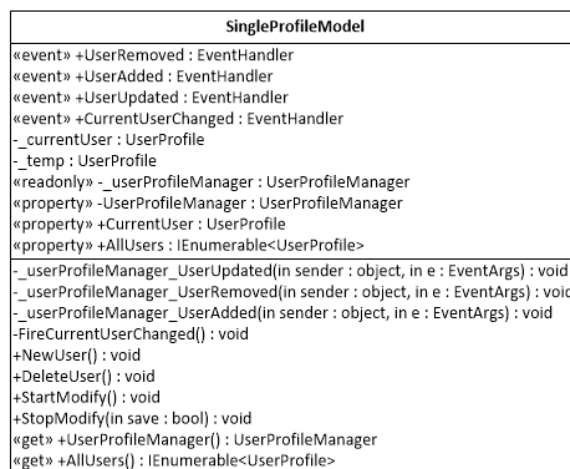


DietCalculator:



Model da associare ai presenter:

Si è voluto mantenere una corretta separazione tra il modello sottostante dei profili utente ed il modello a singolo utente effettivamente utilizzato dall'interfaccia grafica. Per questo motivo abbiamo progettato un'ulteriore classe che funga da *wrapper* attorno al model e aggiunga metodi di comodo per la gestione del singolo utente.

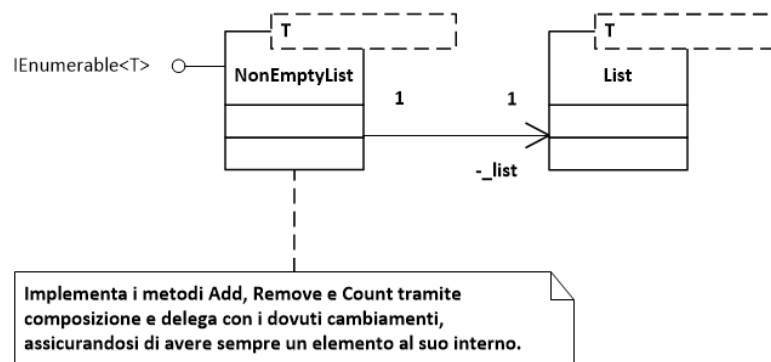


Un contenitore custom per elenchi mai vuoti:

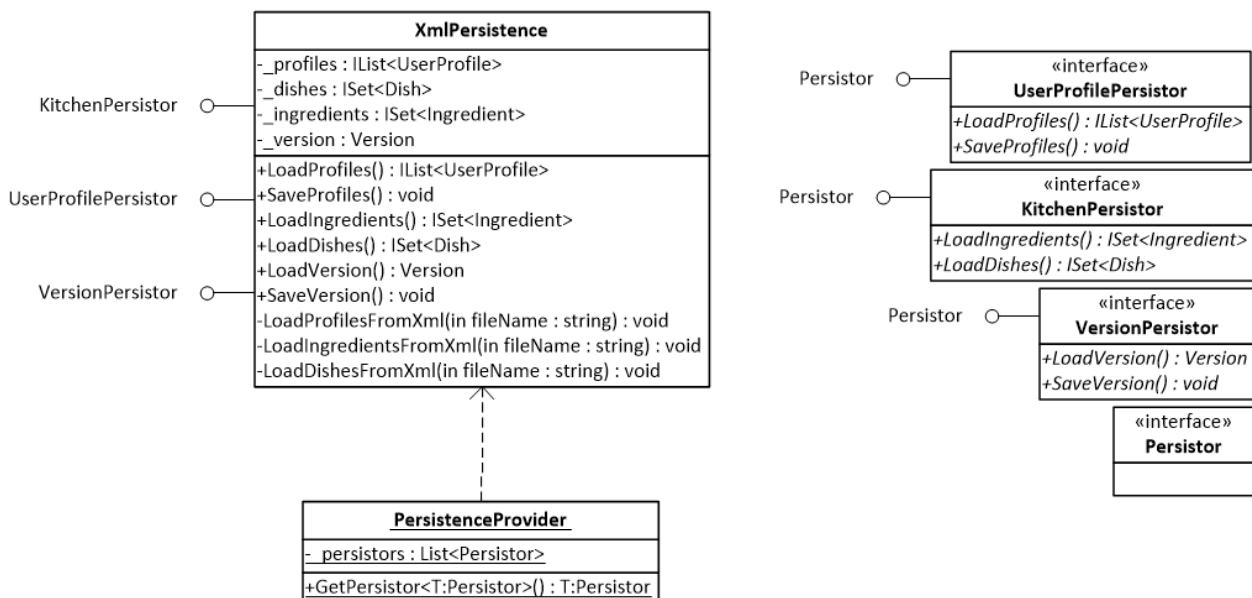
Vista la presenza nel nostro modello di associazioni di cardinalità minima 1 da rispettare (1..*), abbiamo deciso di non utilizzare semplici liste come contenitori. Queste infatti permetterebbero di azzerare il numero di elementi al loro interno violando i vincoli sulle nostre relazioni.

Abbiamo perciò realizzato un contenitore custom che modelli le nostre esigenze, implementando solamente l'interfaccia `IEnumerable<T>` invece che la più scontata `ICollection<T>` per non violare il principio di sostituibilità di Liskov.

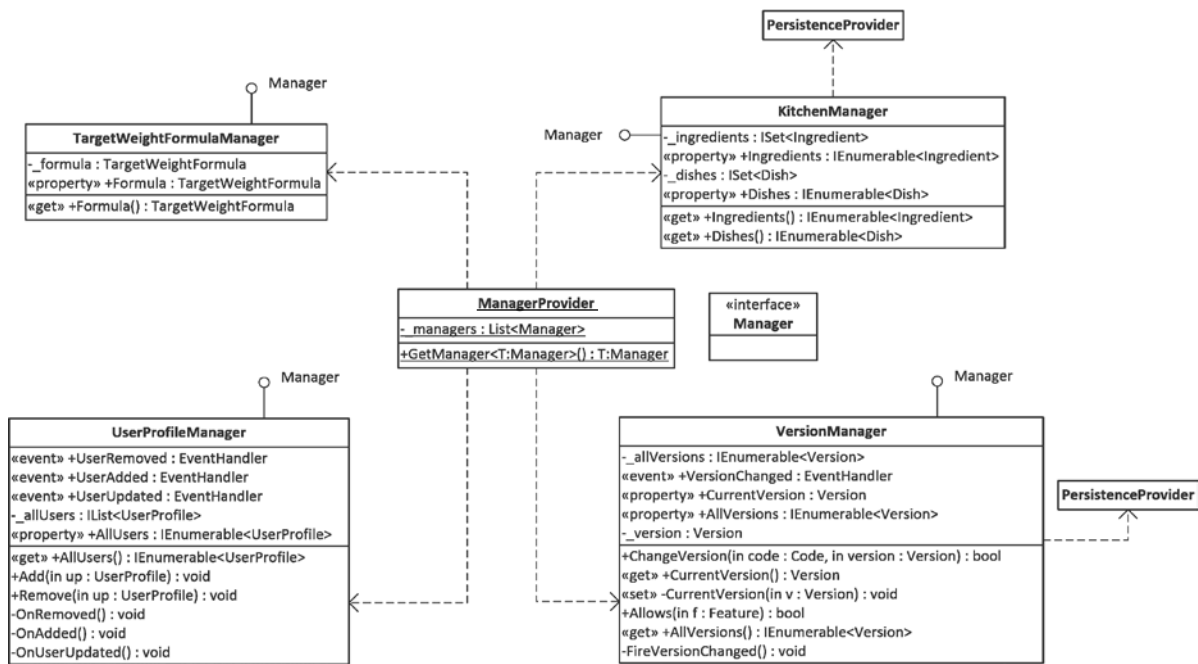
Data la necessità di ottenere un comportamento simile a un contenitore già esistente, nascondendone tuttavia alcuni metodi e modificandone il comportamento di altri, abbiamo incapsulato una `List<T>` e l'abbiamo utilizzata tramite composizione e delega.



Le interfacce per la Persistenza e un'implementazione tramite XML:

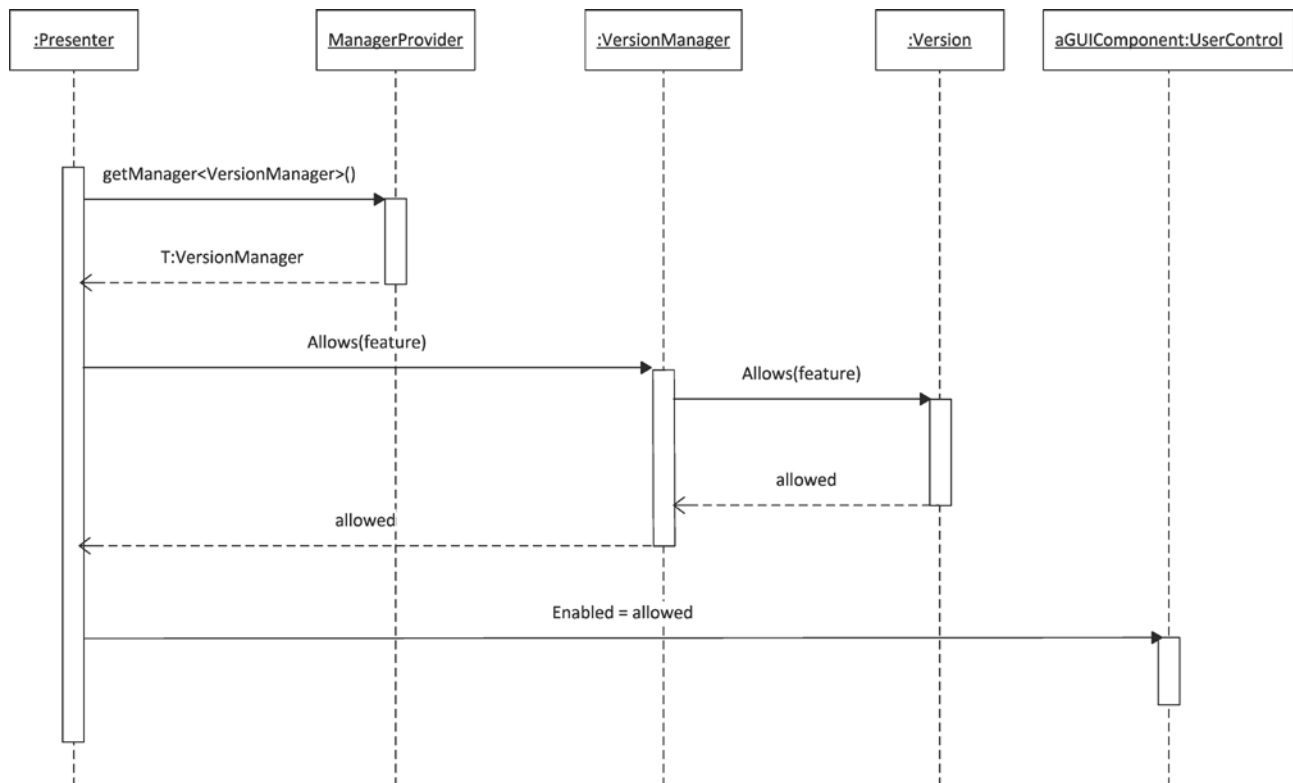


I Manager per la Business Logic:



Modello dinamico:

Diagramma di sequenza:



Con questo diagramma si vuole descrivere come una operazione, associata ad un componente della grafica (`aGUIComponent`), venga abilitata o meno sulla base della versione corrente dell'applicativo.

In particolare si può notare come il presenter ottenga l'implementazione di `VersionManager` a partire da una classe di supporto (`ManagerProvider`). `VersionManager` offre i metodi necessari a validare l'uso delle funzionalità premium del software, appoggiandosi all'istanza della versione corrente. Il risultato della validazione è una variabile booleana che permette quindi di abilitare o meno il componente visuale associato alla feature richiesta.

Principi e pattern di progettazione

Anche i peperoni sono ingredienti “leggeri”

Un primo pattern progettuale utilizzato nel progetto è il *flyweight*, dedicato alla gestione degli ingredienti. Data la loro natura “leggera” (di fatto sono *wrapper* di stringhe), la mancanza di uno stato intrinseco e la possibilità che più ingredienti afferiscano a più piatti, si è deciso di sfruttare il suddetto pattern per dividerli all’interno del sistema. L’implementazione è affidata ad un Set di Ingredient già creati, unitamente a un cast esplicito da string che aggiunge ingredienti al set al momento della creazione.

La strategia di calcolo del peso forma

Per il calcolo del peso forma è stato fatto ricorso al pattern *strategy*. In questo modo il modello dei dati dipende unicamente da un’astrazione (che espone il metodo astratto di calcolo), ed è indipendente dall’implementazione concreta delle varie strategie di calcolo. Si è poi realizzata una classe che specializza tale interfaccia e che fornisce l’implementazione della formula di Keyes (come da documento dei requisiti). Ulteriori implementazioni relative a formule differenti potranno essere successivamente aggiunte alla gerarchia e utilizzate per diversificare il risultato del calcolo, il tutto in maniera perfettamente trasparente all’utente e senza modificare il codice di modello e viste.

La versione è lo stato del sistema

Per ciò che riguarda la gestione delle caratteristiche ammesse o meno dalla versione in uso, si è deciso di subordinare ogni operazione “premium” ad una verifica di applicabilità basata sulla versione attuale del software demandata all’oggetto VersionManager. Il gestore della versione deve poter cambiare dinamicamente comportamento a seguito di ogni operazione di upgrade, adattandosi ogni volta alla nuova versione. Per tali motivi è stato sfruttato il pattern *state* in modo da aggiornare a runtime il riferimento alla Version corrente mantenuto all’interno del manager, al quale esso poi delegherà ogni richiesta di permesso.

Il caricamento di diete da altri Assembly

Tutti gli algoritmi di dieta sono accessibili tramite un metodo factory, in modo tale da non cablare nel codice ogni loro invocazione. Inoltre, le singole classi dalle quali istanziare ogni algoritmo sono recuperate tramite Reflection fra le dll aggiunte al progetto (rispettando così la specifica del documento dei requisiti che imponeva l’indipendenza del software dagli algoritmi di dieta). Per permettere a un altro assembly di realizzare degli algoritmi di calcolo della dieta è stato necessario inserire nel nostro assembly l’interfaccia di un generico algoritmo. In questo modo l’altro progetto avrà una dipendenza a compile time nei confronti del nostro assembly, mentre l’applicativo vero e proprio avrà bisogno delle dll solamente a runtime e tramite l’interfaccia sarà in grado di utilizzare istanze di classi che a compile time non erano note.

Questa scelta progettuale deriva dall’applicazione del principio *open/close* che grazie ad un’interfaccia ben definita, immutabile e per questo “chiusa” permette una grande libertà di estensione e implementazione.

Architettura dell’interfaccia grafica

Infine, la progettazione dell’interfaccia grafica del prototipo è portata avanti con in mente la tecnica *MVP con view passiva*, al fine di rendere le viste totalmente indipendenti dal modello dei dati sottostante. Agendo in questo modo, si ottiene infatti un forte disaccoppiamento tra modello dei dati e view, dato dall’idea che la view si rimetta passivamente al presenter per essere disegnata e il modello possa essere acceduto solamente dal presenter e che ogni cambiamento del modello scateni eventi ricevuti dal presenter. Nell’interfaccia grafica inoltre è stata usata di frequente la Reflection al fine di popolare i componenti con un grado maggiore di libertà e flessibilità.

Le factory di Manager e Persistor

I componenti del sistema che si occupano della gestione del modello (profili utente, repository di ingredienti e piatti ecc.) e della sua persistenza sono stati progettati per essere istanziati e resi accessibili tramite un provider di servizi che si comporta come factory. In questo modo siamo in grado di accedere facilmente al modello da qualsiasi punto del sistema, evitando però di istanziare classi replicate.

Nella progettazione delle interfacce legate ai compiti di managing e persistence è stato tenuto a mente il principio di segregazione delle interfacce, preferendo dunque suddividere diversi metodi all'interno di diverse interfacce per garantire maggiore semplicità di utilizzo da parte di clienti e migliore facilità di manutenzione in caso di modifiche a un'interfaccia. In particolare sono state tenute separate le interfacce relative al caricamento della versione, al caricamento della formula del peso forma, al caricamento dei profili utente e al caricamento di piatti e ingredienti.

Singola responsabilità e inversione delle dipendenze

Nel definire le macro componenti che costituiscono il nostro sistema abbiamo voluto isolare le responsabilità e i compiti associati a ciascuna di esse. In particolare:

- La View ha come unico compito quello di disegnarsi sulla base di ciò che comanda il Presenter
- Il Presenter non si occupa né di grafica né di business logic, ma solamente di fungere da collegamento tra gli eventi della grafica e le conseguenti azioni sul modello e viceversa riflettere sulla view i cambiamenti del modello
- I Manager si occupano della business logic legata alla gestione del modello, ignorando la parte grafica e di interazione con l'utente
- I Persistor hanno l'unico compito di rendere persistenti e ricostruire dal disco o da altre sorgenti le informazioni del modello

Dove possibile (e dove sensato) è stata introdotta una barriera di interfacce tra questi strati applicativi, in modo tale che ogni modulo non dipenda dall'implementazione concreta degli altri, permettendo un maggiore disaccoppiamento e quindi maggiore libertà nel progettarne il cambiamento.

A titolo di esempio, la Persistenza è stata inizialmente implementata da una classe di *mock* che restituiva oggetti creati sul momento, per poi essere sostituita da una persistenza basata su XML una volta raggiunto un certo livello di maturità del prodotto. Nonostante il radicale cambiamento implementativo gli altri componenti non ne hanno risentito perché dipendenti dalle interfacce che astraggono il livello di persistenza.

Inoltre, grazie all'applicazione del principio di segregazione delle interfacce (KitchenPersistor è concettualmente separata da altre interfacce di persistenza come VersionPersistor o UserProfilePersistor) ci è stato possibile cambiare la classe implementativa delle interfacce un "pezzo" alla volta durante la migrazione da *mock* a XML-based, implementando prima la parte più semplice di ingredienti e piatti, per poi dedicarci in un secondo momento ai profili utente. Dopo il processo di migrazione, la persistenza XML è realizzata da un'unica classe che implementa tutte le interfacce di persistenza. Tuttavia grazie alle scelte fatte sarebbe possibile suddividere le responsabilità di questa classe in classi più piccole senza ripercussioni sui componenti che dipendono dalla persistenza. Si potrebbe ad esempio approssicare la gestione degli user profile tramite database locali e con un'altra classe gestire la persistenza della versione con un meccanismo di autenticazione internet-based.