Diet Creator

Progetto del corso

Programmazione ad Oggetti

664426 - Simone Avanzato

Contents

1 Analisi

1.1 Requisiti

L'obiettivo è la realizzazione di una applicazione che permetta all'utente di creare e visualizzare diete personalizzate. Il software gestirà le diete, i vari profili utente, gli alimenti e fornirà le linee guida in fase di creazione della dieta.

1.2 Funzionalità

L'applicazione permetterà di creare, modificare e cancellare profili utente. Ogni utente potrà inserire, modificare e cancellare alimenti e relativi valori nutrizionali. Per poter creare una dieta sarà necessario selezionare un profilo, scegliere un obiettivo e in caso specificare le calorie da aggiungere o da sottrarre per raggiungere tale obiettivo. Dopodiché si dovranno creare i pasti che la compongono inserendo i vari alimenti e le relative quantità. Le diete di un profilo si potranno visualizzare ed eliminare.

Il software esporrà le seguenti funzionalità:

-aggiungere/ modificare/ eliminare profili

-aggiungere/ modificare/ eliminare alimenti con relativi valori nutrizionali

-comporre/ modificare/ eliminare diete

-aggiungere/ eliminare i pasti di una dieta

-visualizzare profili, alimenti e diete

1.3 Modello del dominio

Ogni profilo può avere più diete. Ogni dieta si compone di pasti, ha un target giornaliero di calorie e dei valori minimi e massimi riguardanti la quantità giornaliera di ogni macronutriente. I pasti si compongono di alimenti. Ogni alimento ha dei valori nutrizionali (per 100g), e in base alla quantità inserita in un pasto questi valori variano. Ogni alimento in un pasto dovrà far quindi riferimento ai valori nutrizionali.

2 Progettazione

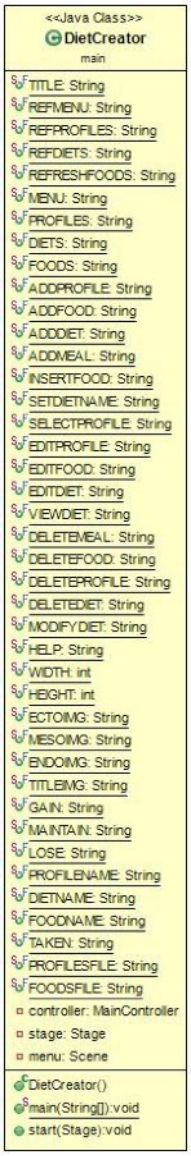
Durante lo sviluppo si è scelto di utilizzare il pattern Model-View-Control per ottenere una chiara suddivisione dei compiti.

2.1 Organizzazione dei package

Il progetto è costituito da quattro package che verranno illustrati in dettaglio qui di seguito.

2.2.1 Package main

Il package main contiene solamente la classe DietCreator, contenente il metodo main.

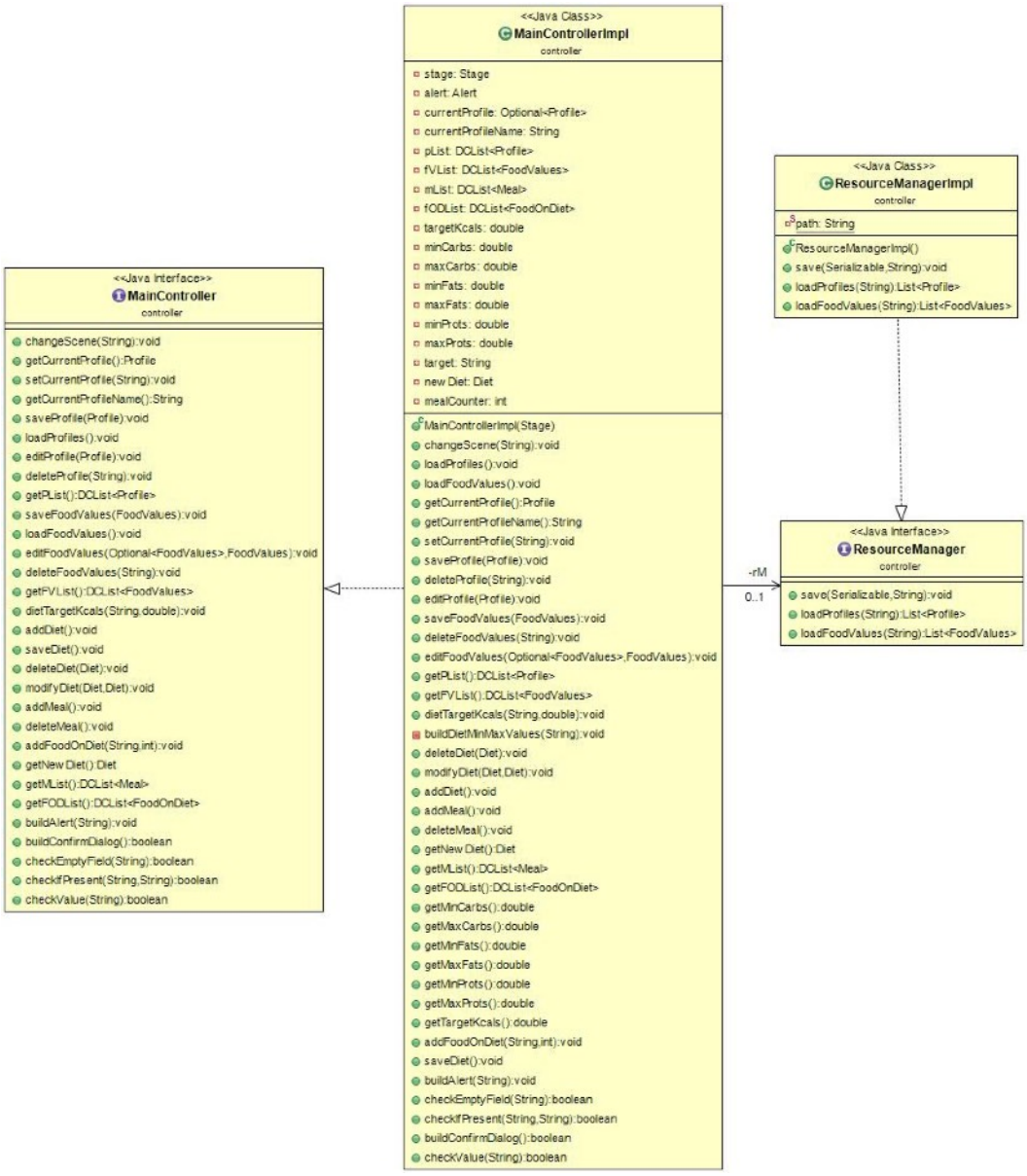


DietCreator

Questa classe è la principale dell'applicazione in quanto la prima ad essere avviata. Contiene lo stage principale e avvia e mantiene l'unica istanza della classe MainController. Il suo metodo start(Stage) inizializza i vari elementi.

2.2.2 Package controller

Questo package contiene le interfacce ResourceManager e MainController e le classi che le implementano ResourceManagerImpl e MainControllerImpl.



MainController

E' l'interfaccia che dichiara i metodi del controller dell'applicazione che fa da punto di raccordo tra l'elemento view e l'elemento model del pattern MVC.

MainControllerImpl

Implementa l'interfaccia MainController. Contiene un riferimento allo stage principale, un riferimento al profilo corrente scelto dall'utente, le liste di profili, diete, pasti e alimenti dell'applicazione e un oggetto ResourceManager. Il metodo changeScene(String) permette di cambiare la scene dello stage principale o di aprire nuovi stage a seconda della necessità. I metodi saveProfile(Profile), editProfile(Profile), deleteProfile(String) permettono di salvare un profilo, modificarlo e cancellarlo. I metodi saveFoodValues(FoodValues), editFoodValues(Optional<FoodValues>, FoodValues), deleteFoodValues(String) servono a salvare un cibo e i relativi valori nutrizionali, modificarlo e cancellarlo. I metodi addDiet(), saveDiet(), deleteDiet(Diet), modifyDiet(Diet, Diet), addMeal(), deleteMeal(), addFoodOnDiet(String, int) sono utilizzati per creare una dieta e i relativi pasti e alimenti che la compongono, oltre alla possibilità di salvataggio, calcallazione e modifica. I metodi buildAlert(String), buildConfirmDialog() servono a creare finestre di avviso in caso di errori o per chiedere conferma dell'azione che si sta per eseguire, ad esempio un cancellazione. I metodi checkEmptyField(String), checkIfPresent(String, String) e checkValue(String) servono per i controlli di correttezza sui campi delle form per la creazione o modifica di profili, diete o alimenti. Vi sono inoltre i vari getter delle liste di oggetti dell'applicazione.

ResourceManager

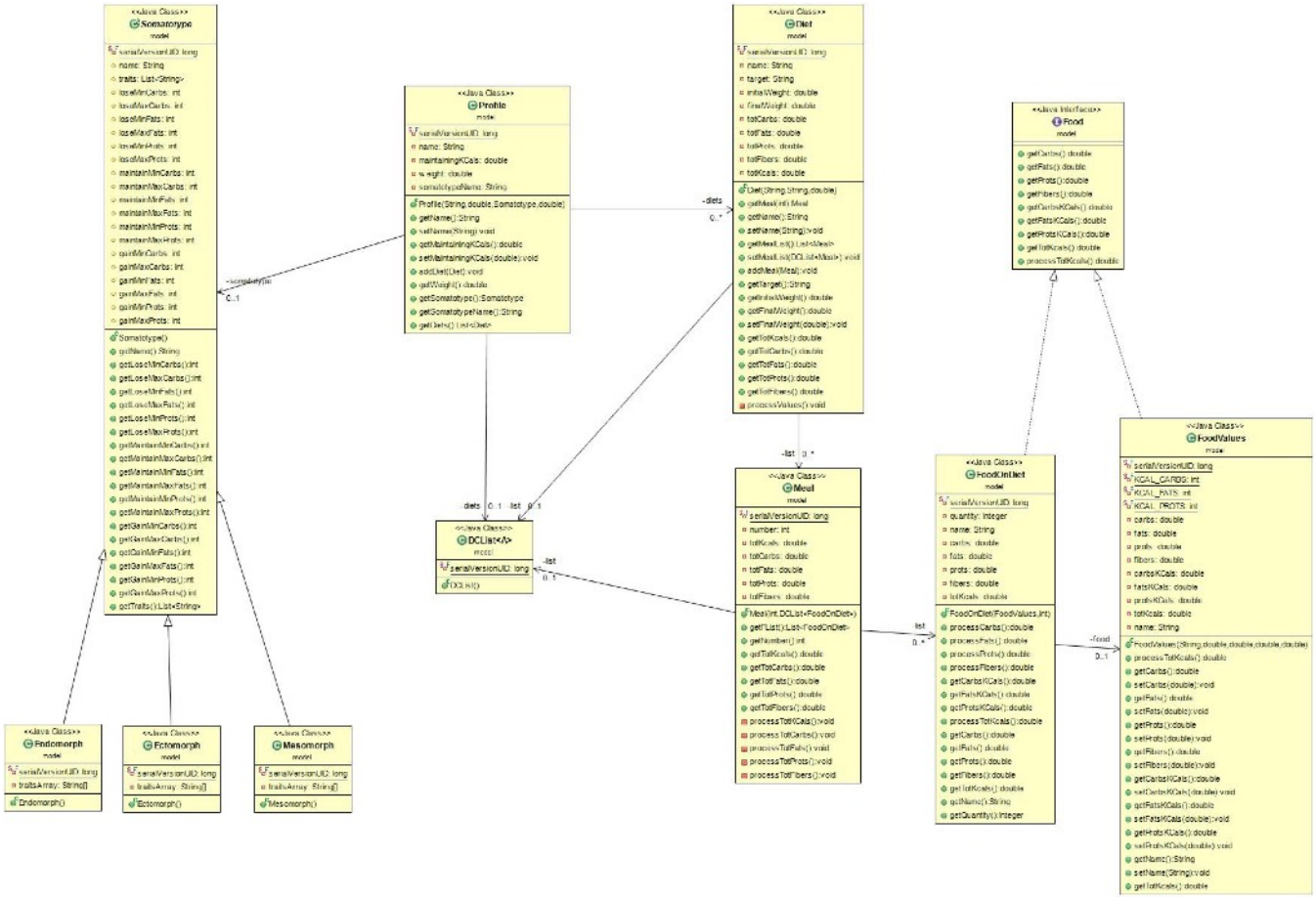
E' l'interfaccia che dichiara i metodi per il salvataggio e caricamento dei dati dell'applicazione.

ResourceManagerImpl

Implementa i metodi dell'interfaccia ResourceManager. Il metodo save(Serializable, String) esegue in caso di successo il salvataggio dell'oggetto passato su un file. I metodi loadProfiles(String), loadFoodValues(String) servono a caricare rispettivamente i profili salvati e gli alimenti salvati in precedenza.

2.2.3 Package model

Questo package contiene l'interfaccia Food e le classi che la implementano FoodValues e FoodOnDiet, la classe astratta Somatotype e le sue estensioni Ectomorph, Mesomorph ed Endomorph, le classi DCList, Meal, Diet e Profile.



Food

Dichiara i metodi getCarbs(), getFats(), getProts(), getFibers(), getCarbsKcals() , getFatsKcals(), getProtsKcals() e getTotKcals() che permettono di ottenere le quantità dei tre macronutrienti e di fibre contenuti in un alimento, le calorie prodotte dai tre macronutrienti e le calorie totali. Il metodo processTotKcals() calcola le calorie totali dell'alimento.

FoodValues

Implementa le interfacce Food e serializable. Aggiunge i metodi setter per i vari valori e un attributo name che ne indica il nome. Rappresenta i valori nutrizionali per 100g di alimento.

FoodOnDiet

Implementa le interfacce Food e serializable. Ha un attributo name e un attributo quantità che ne indica la quantità inserita in un pasto. Ha anche un riferimento a un oggetto FoodValues. Aggiunge i metodi processCarbs(), processFats(), processProts(), processFibers() che calcolano in base alla quantità e al cibo a cui fa riferimento la quantità di ogni macronutriente e di fibre nella porzione da inserire. Rappresenta un alimento inserito in un singolo pasto.

Somatotype

E' una classe astratta che indica il somatotipo di una persona. Ha un attributo name e una lista di tratti che servono a caratterizzarla. Vi sono attributi che indicano la quantità minima e massima di ogni macronutriente a seconda che l'obiettivo sia la perdita, l'aumento o il mantenimento di peso. Ognuno di questi attributi ha un getter.

Ectomorph

Estende la classe astratta somatotype. Rappresenta il somatotipo ectomorfo.

Mesomorph

Estende la classe astratta somatotype. Rappresenta il somatotipo mesomorfo.

Endomorph

Estende la classe astratta somatotype. Rappresenta il somatotipo endomorfo.

Diet

Classe che rappresenta una dieta. Ha un nome, obiettivo, peso iniziale, peso finale e i valori di nutrienti e calorie. Ha una serie di pasti. Il metodo processValues() serve a calcolare il totale di carboidrati, grassi, proteine, fibre e calorie.

Meal

Rappresenta un pasto. Possiede una lista di alimenti e un totale di ogni valore nutrizionale. I metodi processTotCarbs(), processTotFats(), processTotProts(), processTotFibers(), processTotKcals() servono per calcolare questi valori in relazione agli alimenti inseriti.

DCList

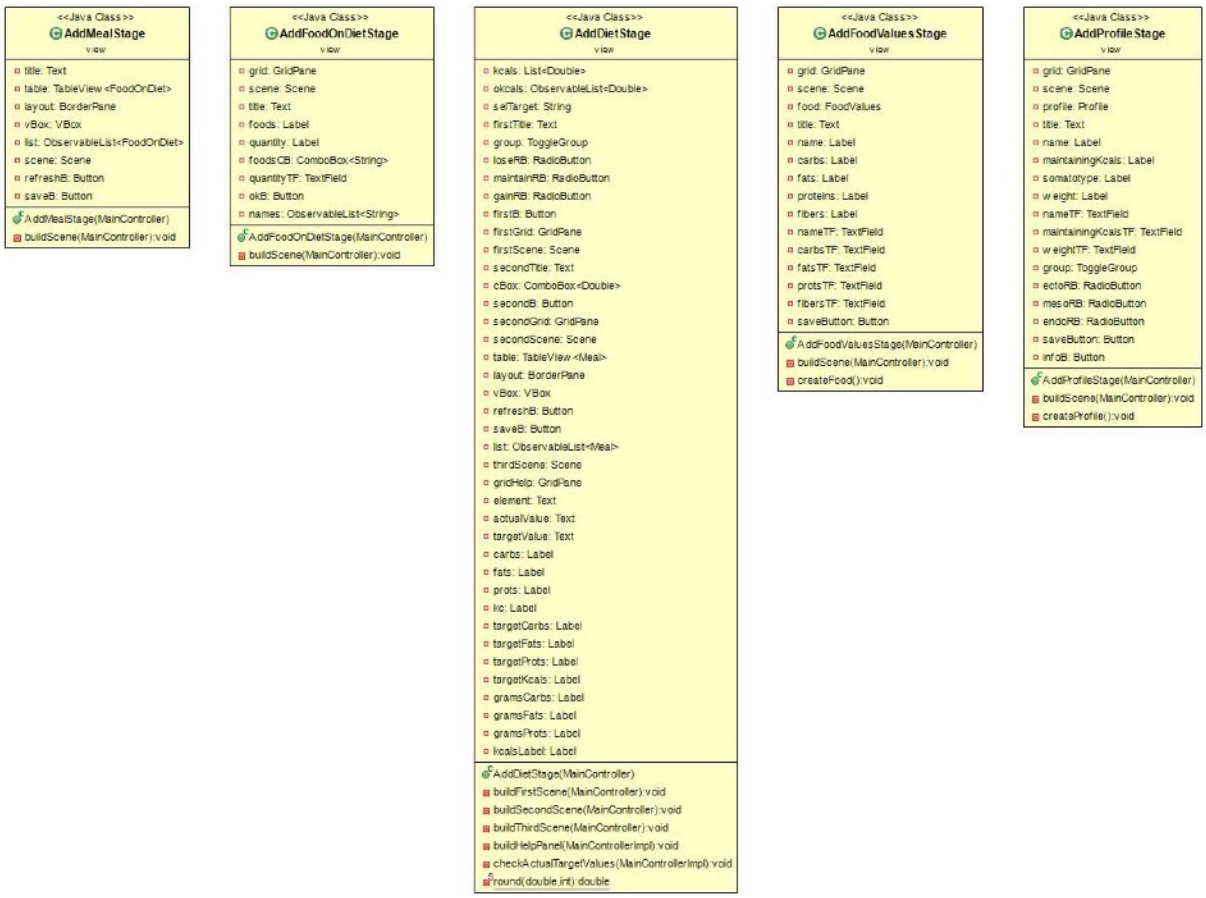
Implementa Serializable e serve fa da lista per i vari elementi dell'applicazione.

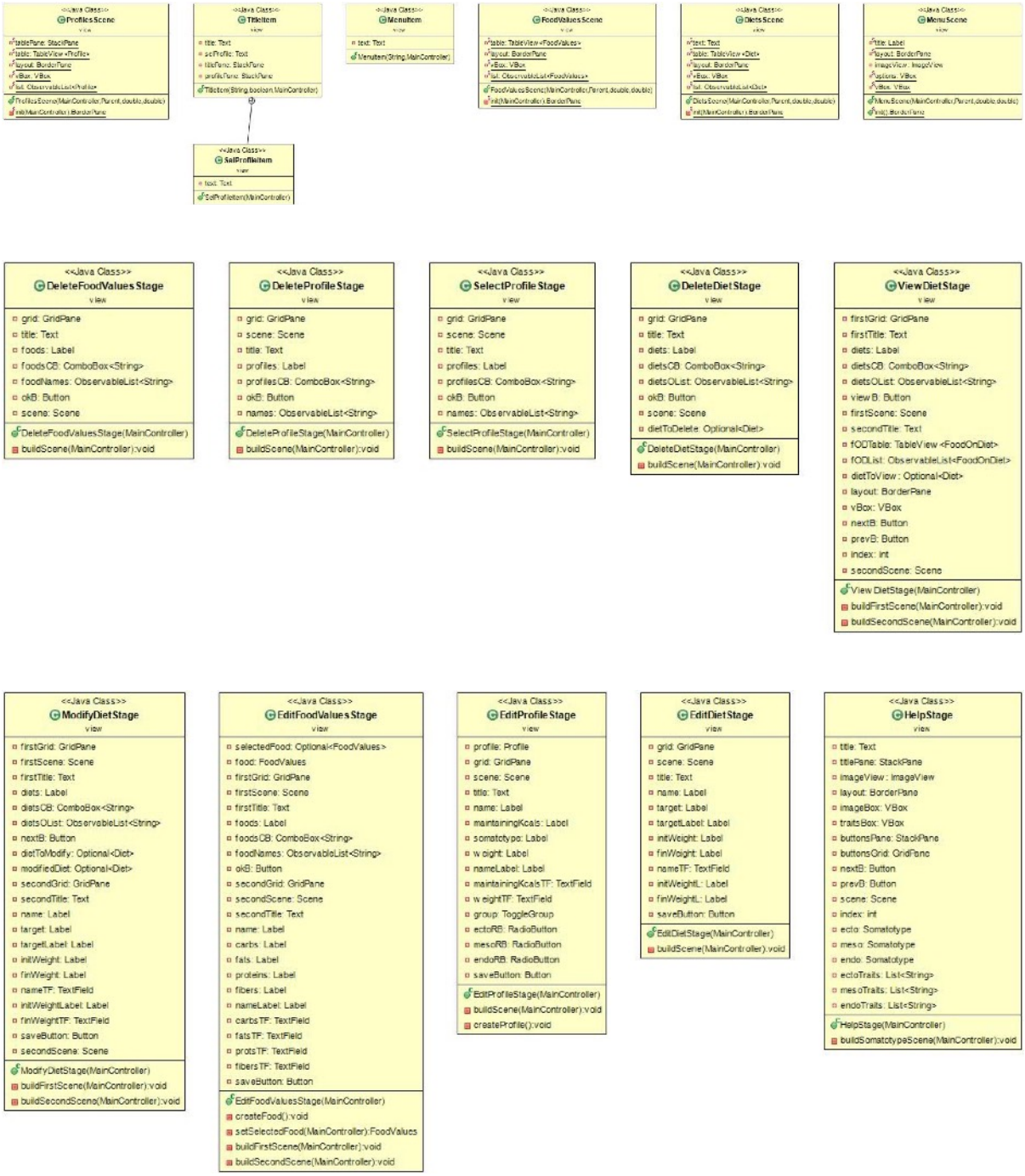
Profile

Rappresenta un profilo utente. Ogni profilo può avere una lista di diete precedentemente compilate e salvate.

2.2.3 Package view

Questo package contiene le scene principali dell'applicazione, cioè le classi ProfilesScene, FoodValuesScene, DietScene e MenuScene. Vi sono inoltre le finestre aggiuntive, cioè le classi AddDietStage, AddFoodOnDietStage, AddFoodValuesStage, AddMealStage, AddProfileStage, DeleteDietStage, DeleteFoodStage, DeleteProfileStage, EditDietStage, EditFoodStage, EditProfileStage, HelpStage, ModifyDietStage, SelectProfileStage, ViewDietStage e le classi MenuItem e TitleItem.





MenuScene

Estende la classe Scene e rappresenta il menu del programma. Ha un metodo init() che serve a inizializzare i vari elementi grafici. Il riferimento al controller è passato tramite costruttore.

ProfilesScene

Estende la classe Scene e rappresenta la sezione profili dell'applicazione. Ha un metodo init() che serve a inizializzare i vari elementi. Espone una tabella che illustra tutti i profili utente. Il riferimento al controller è passato tramite costruttore.

DietsScene

Estende la classe Scene e rappresenta la sezione diete. Ha un metodo init() per inizializzare i vari elementi e illustra tramite una tabella le varie diete appartenenti all'utente corrente. Il riferimento al controller è passato tramite costruttore.

FoodValuesScene

Estende la classe Scene e rappresenta la sezione alimenti dell'applicazione. Ha un metodo init() che serve a inizializzare i vari elementi. Ha una tabella che rappresenta gli alimenti salvati e i relativi valori nutrizionali per 100g di alimento. Il riferimento al controller è passato tramite costruttore.

AddDietStage

Estende la classe stage. E' a finestra che permette all'utente di creare una dieta. Si compone di tre scene inizializzate dai metodi buildFirstScene(MainController), buildSecondScene(MainController) e buildThirdScene(MainController).
Il metodo buildHelpPanel(MainController) inizializza il pannello che servirà all'utente come guida alla compilazione della dieta. Il metodo checkActualTargetValues(MainController) serve a calcolare i valori obiettivo della dieta e quelli attuali. Il metodo round(double, int) serve ad arrotondare i valori numerici. Il riferimento al controller è passato tramite costruttore.

AddFoodOnDietStage

Estende la classe Stage. Permette di aggiungere un alimento ad un pasto della dieta che si sta compilando.
Il metodo buildScene(MainController) inizializza gli elementi della finestra. Il riferimento al controller è passato tramite costruttore.

AddFoodValuesStage

Estende la classe Stage. Il riferimento al controller è passato tramite il costruttore. Permette all'utente di creare ed inserire un alimento ed

i relativi valori nutrizionali. Il metodo createFood() crea il nuovo alimento e lo passa al controller.

AddMealStage

Estende la classe Stage. Il riferimento al controller è passato tramite il costruttore. Serve a creare un pasto da aggiungere in seguito ad una dieta che si sta compilando.
Il metodo buildScene(MainController) inizializza i vari elementi.

AddProfileStage

Estende la classe Stage. Il riferimento al controller è passato tramite il costruttore. Tramite questa finestra l'utente può creare un profilo e salvarlo. Il metodo createProfile() serve a creare un profilo nuovo che sarà poi passato al controller in caso di salvataggio.

DeleteDietStage

Estende la classe Stage. Permette all'utente di selezionare ed eliminare una dieta del profilo corrente. Il metodo buildScene() inizializza i vari elementi. Il riferimento al controller è passato tramite il costruttore.

DeleteFoodStage

Estende la classe Stage. Il riferimento al controller è passato dal costruttore. Serve a selezionare ed eliminare un alimento. Il metodo buildScene() inizializza gli elementi.

DeleteProfileStage

Estende la classe Stage. L'utente può selezionare ed eliminare un profilo utente. Il metodo buildScene() serve ad inizializzare tutti gli elementi della classe. Il controllore viene passato tramite costruttore.

ModifyDietStage

Estende la classe Stage. Il controllore è passato tramite costruttore. Serve all'utente per modificare nome o aggiornare il peso finale di una dieta. Gli elementi sono inizializzati dal metodo buildScene().

EditFoodStage

Estende la classe Stage. Il controllore è passato tramite costruttore. L'utente può modificare i vari valori nutrizionali di un alimento. Gli elementi sono inizializzati dal metodo buildScene().

EditProfileStage

Estende la classe Stage. Permette di modificare alcuni dati di un profilo utente. Gli elementi vengono inizializzati dal metodo buildScene().

EditDietStage

Estende la classe Stage. Il controllore è passato tramite costruttore. Serve all'utente per cambiare il nome di una dieta che sta compilando. Gli elementi sono inizializzati dal metodo buildScene().

SelectProfileStage

Estende la classe Stage. Il controllore viene passato tramite costruttore. Permette di selezionare il profilo utente da utilizzare.

HelpStage

Estende la classe Stage. E' una finestra di aiuto per la scelta del somatotipo durante la creazione di un nuovo profilo. Gli elementi sono inizializzati dal metodo buildSomatotypeScene(). Il controllore è passato tramite costruttore.

TitleItem

Estende la classe VBox e rappresenta il titolo di diverse scene e mostra l'attuale profilo selezionato. Contiene la classe innestata SelProfileItem. Il riferimento al controller viene passato tramite costruttore.

SelProfileItem

E' una classe innestata di TitleItem. E' l'elemento grafico che permette di selezionare o cambiare profilo.

MenuItem

Estende la classe StackPane e rappresenta una voce menu. Il riferimento al controller è passato tramite costruttore.

3 Implementazione

Dopo la fase di progettazione ha avuto inizio la realizzazione dell'applicazione.

3.1 File dei profili e File degli alimenti

Per il salvataggio dei profili e degli alimenti si è scelto di utilizzare una lista. Il file viene salvato in una cartella sul desktop. Se tale cartella non esiste, verrà creata dall'applicazione insieme al file.

3.2 Logica dell'applicazione

L'utente utilizza l'applicazione per poter creare e consultare in futuro diete che gli consentano di raggiungere un obiettivo in termini di peso, seguendo la teoria dei somatotipi. L'utente ha a disposizione un menu e diverse sezioni per creare, gestire e visionare profili, diete e alimenti.
L'applicazione fornisce nella fase centrale, ovvero la creazione di una dieta, linee guida sullo stato attuale della dieta che si sta costruendo in relazione alle soglie minime e massime che si devono raggiungere per ogni macronutriente. Per questa guida si è scelto di creare un pannello posto nella finestra di creazione di una dieta.

3.3 Interfaccia grafica

Per quanto riguarda l'interfaccia grafica si è scelto di utilizzare la libreria JavaFX.

4 Conclusioni

Lo sviluppo è stato lineare e mi ritengo soddisfatto per quanto riguarda l'implementazione di tutte le funzionalità richieste dall'applicazione. Per motivi di tempo non si è riuscito ad implementare la funzionalità che permettesse di creare una lista della spesa partendo da una dieta. Sono soddisfatto dell'esperienza e al tempo stesso dispiaciuto di non essere riuscito a lavorare in gruppo e provare quindi un diverso tipo di approccio.

5 Guida d'uso

Creazione profili

Per creare un profilo si deve accedere dal menu principale alla sezione "profiles" e poi selezionare la voce "add profile". Nella form "new profile" occorre inserire un nome utente, il peso in kg, il somatotipo e le calorie di mantenimento del peso corporeo (calcolabili con il seguente tool online: http://www.calculator.net/calorie-calculator.html).

Creazione diete

Per creare una dieta occorre prima di tutto aver selezionato un profilo ("select profile"), poi andare nella sezione "diets" e selezionare la voce "add diet". Si dovrà selezionare l'obiettivo della dieta ed eventualmente le calorie in più/in meno per poter raggiungere tale obiettivo. Si aprirà la schermata dieta che presenta in basso a sinistra gli obiettivi in grammi per quanto riguarda i macronutrienti e in cal per quanto riguarda la soglia calorica. Per aggiungere pasti alle diete selezionare la voce "add meal" e per inserire alimenti e la relativa quantità in grammi in un pasto, selezionare "insert food". Per aggiornare la schermata si clicki il pulsante "refresh" posto in basso e centralmente. Occorre inserire un nome per la dieta, selezionando la voce "edit diet".

Inserimento alimenti e valori nutrizionali

Per creare ed inserire un alimento e i relativi valori nutrizionali (per 100 grammi di alimento) occorre accedere dal menu principale alla sezione "foods" e poi selezionare "add food". Nella form "New food" inserire il nome dell'alimento e i valori nutrizionali in grammi e kcal.

Note aggiuntive

Nel repository del progetto ho aggiunto due files denominati "profiles.bin" e "foods.bin" che contengono profili e alimenti già inseriti nelle varie prove. Per poterli utilizzare si crei una cartella sul desktop, denominandola "DietCreatorSaves", si copino i file all'interno della suddetta cartelle e si esegua l'applicazione.