



Progetto ingegneria del software

TrackMe



Obiettivo

Lo scopo è di creare un'applicazione dedicata al **monitoraggio** delle calorie consumate dall'utente nel corso della giornata.

Potremmo riassumere i principali obiettivi nei seguenti punti:

- **Responsabilizzare** l'utente nei confronti di un'alimentazione sana e controllata, fornendo strumenti e informazioni utili per raggiungere e mantenere gli obiettivi di salute e benessere.
- Favorire la **consapevolezza** dell'equilibrio calorico e dell'importanza della nutrizione bilanciata.
- Offrire un'esperienza utente **intuitiva** e **personalizzata** per facilitare il monitoraggio dell'assunzione calorica quotidiana.
- **Promuovere** la creazione di abitudini alimentari salutari attraverso suggerimenti, notifiche e obiettivi individualizzati.
- **Integrare** funzionalità aggiuntive, come la registrazione delle attività fisiche e il tracciamento degli obiettivi di perdita o guadagno di peso.
- Fornire un **supporto costante** e **motivazionale** per incoraggiare l'utente nel percorso verso uno stile di vita più sano e attivo.

Difficoltà incontrate

Durante lo sviluppo dell'applicazione, abbiamo affrontato diverse sfide e difficoltà che hanno richiesto tempo e risorse per essere superate. Alcune delle principali difficoltà incontrate includono:

- **Complessità dell'analisi dei requisiti:** Definire in modo **chiaro e completo** i requisiti dell'applicazione ha richiesto un'**attenta analisi** delle esigenze degli utenti e dei processi coinvolti nel **monitoraggio** dell'assunzione calorica.
- **Integrazione dei dati nutrizionali:** Ottenere e integrare dati **accurati e aggiornati** sui valori nutrizionali degli alimenti è stato un compito impegnativo, richiedendo la **ricerca** e l'utilizzo di fonti affidabili e verificabili.
- **Organizzazione degli incontri e comunicazione:** Gestire efficacemente gli **incontri** di team, la **comunicazione** e la **collaborazione** tra i membri del team è stata una sfida importante.



Nonostante queste sfide, il nostro team ha affrontato con determinazione ogni ostacolo e ha lavorato duramente per garantire il successo del progetto.

Difficoltà incontrate

- **Progettazione dell'interfaccia utente:** Creare un'interfaccia utente **intuitiva** e **user-friendly** è stato un processo iterativo, in cui abbiamo dovuto considerare diverse opzioni di **design** e raccogliere feedback per **ottimizzare** l'esperienza complessiva dell'applicazione.
- **Implementazione delle funzionalità avanzate:** Introdurre funzionalità avanzate, come il **calcolo** automatico delle **calorie consumate** e la **generazione** di **report** personalizzati, ha richiesto un'attenta progettazione e sviluppo per garantire la precisione e l'affidabilità dei risultati.
- **Test e debug:** Condurre **test approfonditi** e **individuare** e **risolvere** eventuali **bug** e **problemi** di performance è stato un processo continuo e fondamentale per garantire la **stabilità** e l'**efficienza** dell'applicazione.



Nonostante queste sfide, il nostro team ha affrontato con determinazione ogni ostacolo e ha lavorato duramente per garantire il successo del progetto.

Paradigma di programmazione e strumenti

Nel corso dello sviluppo del progetto, abbiamo adottato diversi strumenti e tecnologie che riflettono il nostro approccio al paradigma di programmazione. Questi strumenti sono stati scelti per la loro efficacia, affidabilità e capacità di supportare il nostro processo di sviluppo. Essi includono:

- **Ambiente di sviluppo integrato (IDE): Eclipse** è stato il nostro principale ambiente di **sviluppo** per la scrittura del codice **Java**. Offre un'ampia gamma di funzionalità e strumenti per facilitare lo sviluppo e la **gestione** del progetto.
- **Gestione del database:** Per la gestione del **database SQLite**, abbiamo utilizzato **DBeaver**, un potente strumento open-source che ci ha permesso di **creare**, **modificare** e **interrogare** il database in modo efficiente.
- **Strumento per la Modellazione UML:** Per la creazione dei **diagrammi UML**, abbiamo impiegato **StarUML**, uno strumento intuitivo e completo che ci ha consentito di **visualizzare** e **progettare** l'architettura del software in modo chiaro e preciso.



L'adozione di questi strumenti ha svolto un ruolo fondamentale nel supportare il nostro processo di sviluppo e nel garantire il successo del progetto.

Paradigma di programmazione e strumenti

- **Piattaforma di hosting:** **GitHub** è stata la nostra piattaforma di **hosting** principale, utilizzata per **condividere** il codice, **gestire** le versioni e **collaborare** con gli altri membri del team.
- **Strumento di presentazione:** Infine, per la creazione della **presentazione** finale, abbiamo utilizzato **Microsoft PowerPoint**, un software ampiamente utilizzato e ben noto per la sua **versatilità** e **facilità** d'uso nella creazione di presentazioni professionali.



L'adozione di questi strumenti ha svolto un ruolo fondamentale nel supportare il nostro processo di sviluppo e nel garantire il successo del progetto.



Software configuration management

Il Software Configuration Management (**SCM**) è una pratica essenziale per garantire una **gestione** efficace e sistematica del **codice** e della **documentazione** del progetto. Nel corso del nostro lavoro, abbiamo adottato diverse strategie e strumenti per gestire la configurazione del software. Alcuni aspetti chiave includono:

- **Utilizzo di GitHub:** Abbiamo utilizzato GitHub come piattaforma di hosting principale per il nostro progetto. GitHub ci ha fornito un ambiente centralizzato per la **condivisione** del codice, la **gestione** delle versioni e la **collaborazione** tra i membri del team.
- **Organizzazione del repository:** Abbiamo strutturato il repository GitHub in modo efficiente, con directory dedicate per il **codice sorgente** e la **documentazione**. Questa organizzazione ci ha aiutato a mantenere il progetto **ordinato** e facilmente **navigabile**.



La corretta gestione della configurazione del software è stata fondamentale per mantenere l'ordine e la coerenza nel nostro progetto e per garantire una collaborazione efficace e produttiva tra i membri del team.



Software configuration management

- **Gestione delle modifiche:** *Prima* di caricare modifiche significative sul repository, abbiamo concordato di *discutere* e *valutare* le modifiche come team. Questo ci ha permesso di evitare errori e garantire che il codice caricato fosse di alta qualità.
- **Organizzazione dei compiti:** Abbiamo utilizzato le sezioni "Issues" e "Projects" di GitHub per tenere traccia dei *compiti* e delle *attività* da svolgere. Ogni problema è stato *assegnato* a un membro del team, facilitando la *collaborazione* e la *suddivisione* del lavoro.
- **Documentazione dettagliata:** Abbiamo mantenuto una documentazione dettagliata all'interno del repository, includendo *guide* per lo sviluppo e *note* sulle decisioni prese.



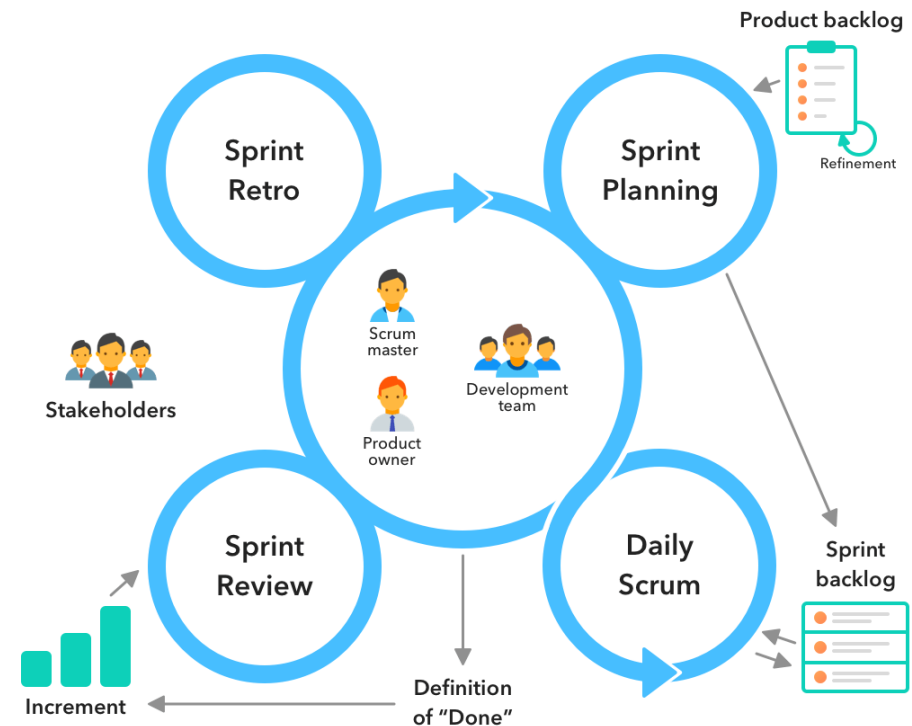
La corretta gestione della configurazione del software è stata fondamentale per mantenere l'ordine e la coerenza nel nostro progetto e per garantire una collaborazione efficace e produttiva tra i membri del team.

SCRUM life cycle

Nel nostro progetto, abbiamo adottato l'approccio organizzativo Scrum, un framework agile ampiamente utilizzato nello sviluppo software. Esso si basa su un processo **iterativo** e **incrementale**, suddividendo il lavoro in brevi periodi di tempo chiamati sprint.

Le caratteristiche principali dell'organizzazione Scrum includono:

- **Squadra Scrum:** Una squadra **auto-organizzata** e multifunzionale responsabile dello sviluppo del prodotto.
- **Sprint:** Periodi di **tempo predeterminati** (solitamente da una a quattro settimane) durante i quali la squadra sviluppa **incrementi** di **funzionalità** utilizzabili del prodotto.
- **Riunioni Scrum:** **Riunioni** regolari, tra cui lo *Sprint Planning*, il *Daily Scrum*, lo *Sprint Review* e la *Sprint Retrospective*, che aiutano a sincronizzare il lavoro della squadra e a riflettere sulle prestazioni.
- **Backlog del prodotto e backlog dello sprint:** **Elenco** di lavori da fare, **prioritizzati** dal valore commerciale e **divisi** in piccoli compiti gestiti durante lo sprint.



L'organizzazione Scrum favorisce la trasparenza, la collaborazione e l'adattamento continuo, consentendo al team di rispondere in modo flessibile ai cambiamenti nei requisiti e alle sfide emergenti.

Requisiti

L'estrazione e la specifica dei requisiti sono stati processi fondamentali per garantire il successo del prodotto. Considerando la natura accademica del progetto, il processo di definizione dei requisiti è stato principalmente **interno** al nostro team di sviluppo. Ecco come abbiamo proceduto:

Estrazione dei requisiti:

- Abbiamo intrapreso sessioni di **brainstorming** interne al team, durante le quali abbiamo discusso e identificato le **esigenze** e le **funzionalità principali** dell'applicazione.
- Ci siamo basati sulle nostre **conoscenze** e sulle **linee guida** fornite dal corso per definire i requisiti funzionali e non funzionali.
- Abbiamo analizzato e discusso **casi d'uso ipotetici** per comprendere come gli utenti avrebbero interagito con l'applicazione e quali funzionalità avrebbero desiderato.
- Sfruttando le nostre competenze tecniche e le conoscenze acquisite nel corso del progetto, siamo stati in grado di individuare i **vincoli** e le **restrizioni** del sistema.

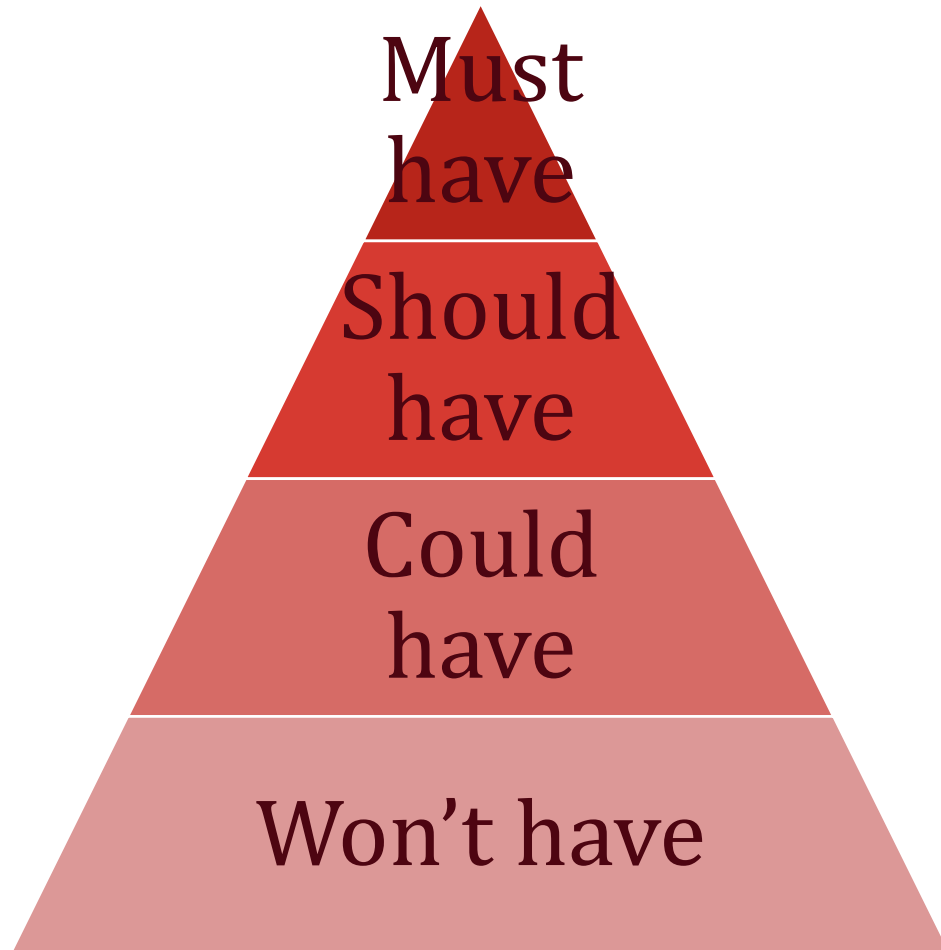
Specifica dei requisiti:

- Abbiamo **documentato** i requisiti in **dettaglio**, utilizzando documenti appositamente creati per il progetto.
- Per visualizzare i requisiti in modo più chiaro, abbiamo utilizzato **diagrammi UML**, adattati alle dimensioni del nostro progetto e alle nostre esigenze specifiche.



Nonostante le limitate interazioni esterne, il processo di definizione dei requisiti è stato una parte cruciale del nostro percorso di sviluppo, garantendo che il prodotto finale rispondesse alle nostre aspettative.

Requisiti



I requisiti del progetto sono stati delineati e dettagliati nel documento della specifica dei requisiti.

In particolare, abbiamo utilizzato due approcci:

Modello MOSCOW

Must-have: requisiti essenziali per il funzionamento base dell'applicazione.

Should-have: requisiti desiderabili ma non indispensabili.

Could-have: requisiti opzionali che possono essere inclusi se il tempo e le risorse lo permettono.

Won't-have: requisiti non inclusi nell'attuale versione del progetto ma considerati per il futuro.

Modello KANO

Requisiti funzionali: riguardano le funzionalità dirette del sistema.

Requisiti non funzionali: definiscono gli standard di qualità, accessibilità e sicurezza da rispettare.

Qualità del software

La qualità del software è un aspetto fondamentale per garantire la **soddisfazione** dell'utente e il **successo** del progetto. I principali attributi di qualità includono:

Funzionamento

- Correttezza
- Affidabilità
- Efficienza
- Integrità
- Usabilità

Revisione

- Manutenibilità
- Testabilità
- Flessibilità

Transizione

- Portabilità
- Riutilizzabilità
- Interoperabilità

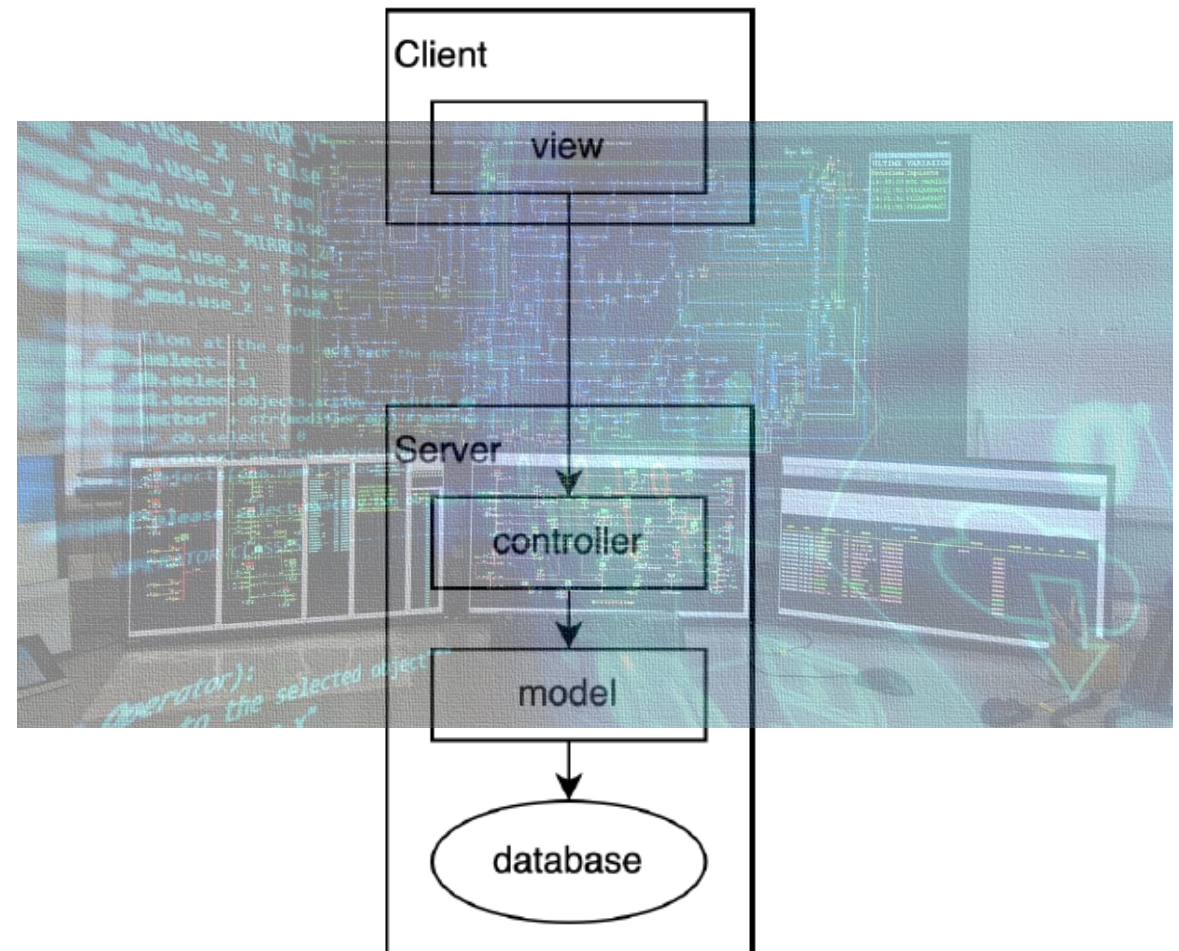


Architettura

L'architettura software definisce la struttura fondamentale del sistema e le relazioni tra i suoi componenti.

Le caratteristiche principali includono:

- **Architettura a tre strati:** Divide il sistema in **tre livelli** distinti: *interfaccia grafica, logica e accesso ai dati*.
- **Scalabilità:** Il sistema può **crescere** per gestire un aumento del carico o delle richieste degli utenti.
- **Flessibilità:** Il sistema è progettato per **adattarsi** facilmente a **cambiamenti** nei requisiti o nell'ambiente.
- **Sicurezza:** Vengono implementati meccanismi di protezione per garantire la **sicurezza** dei **dati** e degli utenti.
- **Prestazioni:** Il sistema è **ottimizzato** per garantire tempi di risposta rapidi e prestazioni efficienti. Un'architettura ben progettata fornisce una base solida per lo **sviluppo** del software e ne facilita l'**evoluzione** e la **manutenzione** nel tempo.



Design patterns

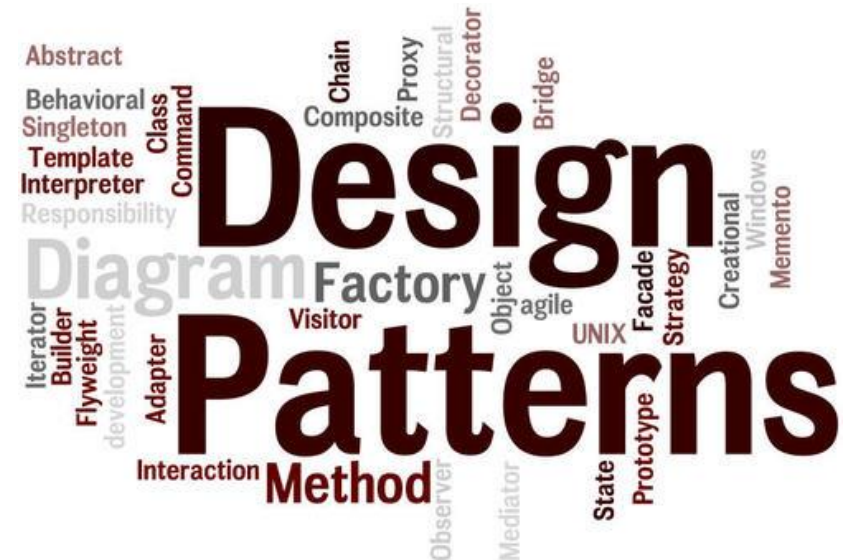
Un design pattern è una metodologia consolidata per **risolvere problemi ricorrenti** nella progettazione del software. Questi modelli offrono **soluzioni provate e ottimizzate** per problemi specifici, facilitando lo sviluppo di software **efficiente e manutenibile**.

Design patterns utilizzati:

- **MVC:** *Model*: Gestisce i dati e le operazioni senza conoscere l'interfaccia utente. *View*: Presenta i dati all'utente in modo comprensibile, senza logica di business. *Controller*: Media tra modello e vista, gestendo gli input e aggiornando lo stato del modello.
- **Singleton**: Per garantire l'esistenza di una sola istanza di una classe e fornire un **punto di accesso globale** ad essa.
- **Factory method**: Per **creare oggetti** senza specificare la classe esatta dell'oggetto che verrà creato.
- **Observer**: Per stabilire una **dipendenza** uno-a-molti tra oggetti, in modo che quando un oggetto cambia stato, tutti i suoi dipendenti vengono notificati e aggiornati automaticamente.

Vantaggi dei design patterns:

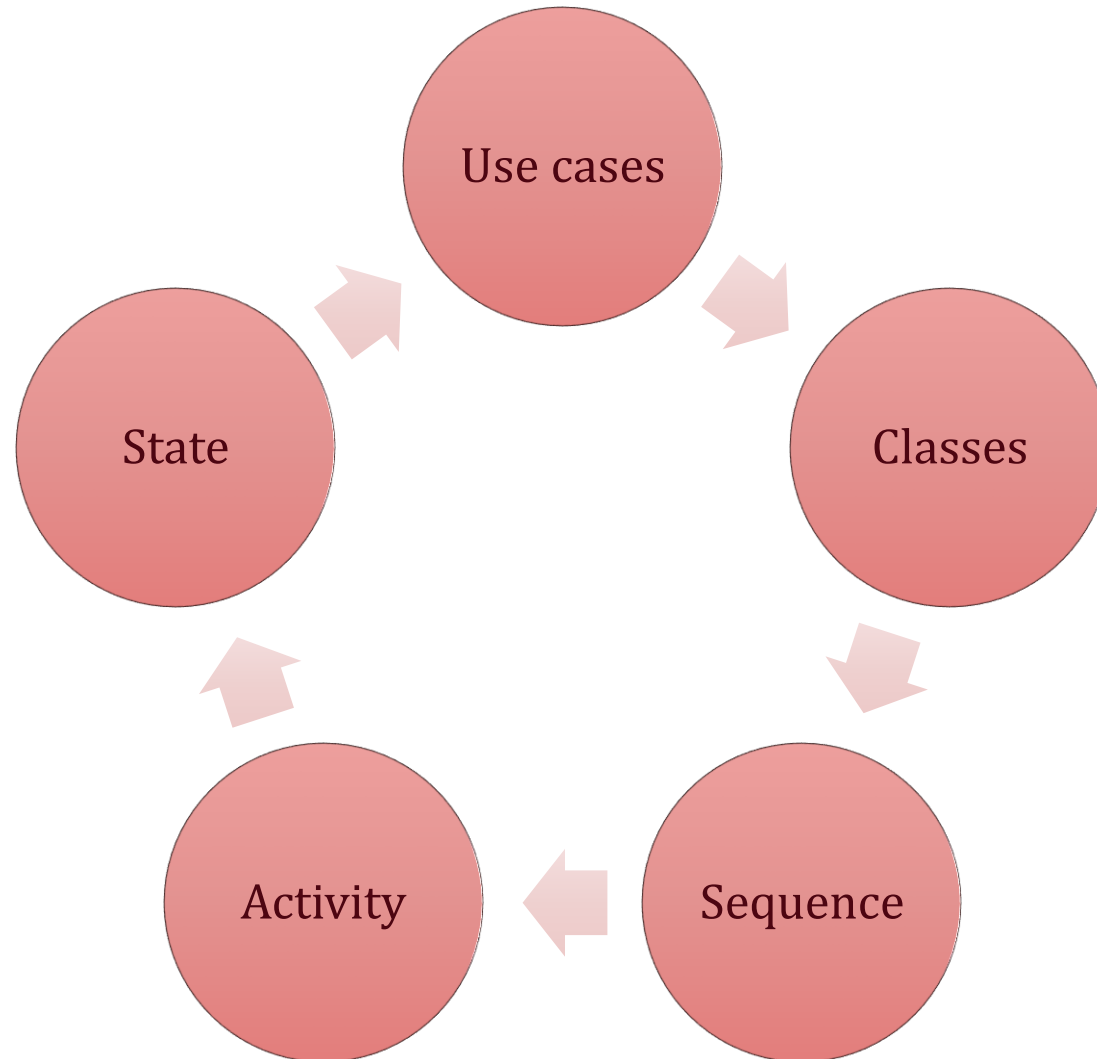
- Promuovono la **riusabilità** del codice e la **manutenibilità**.
- Forniscono una **soluzione comune** a problemi ricorrenti.
- Migliorano la **comprensione** del codice e la **comunicazione** tra i membri del team.





StarUML

Modellazione

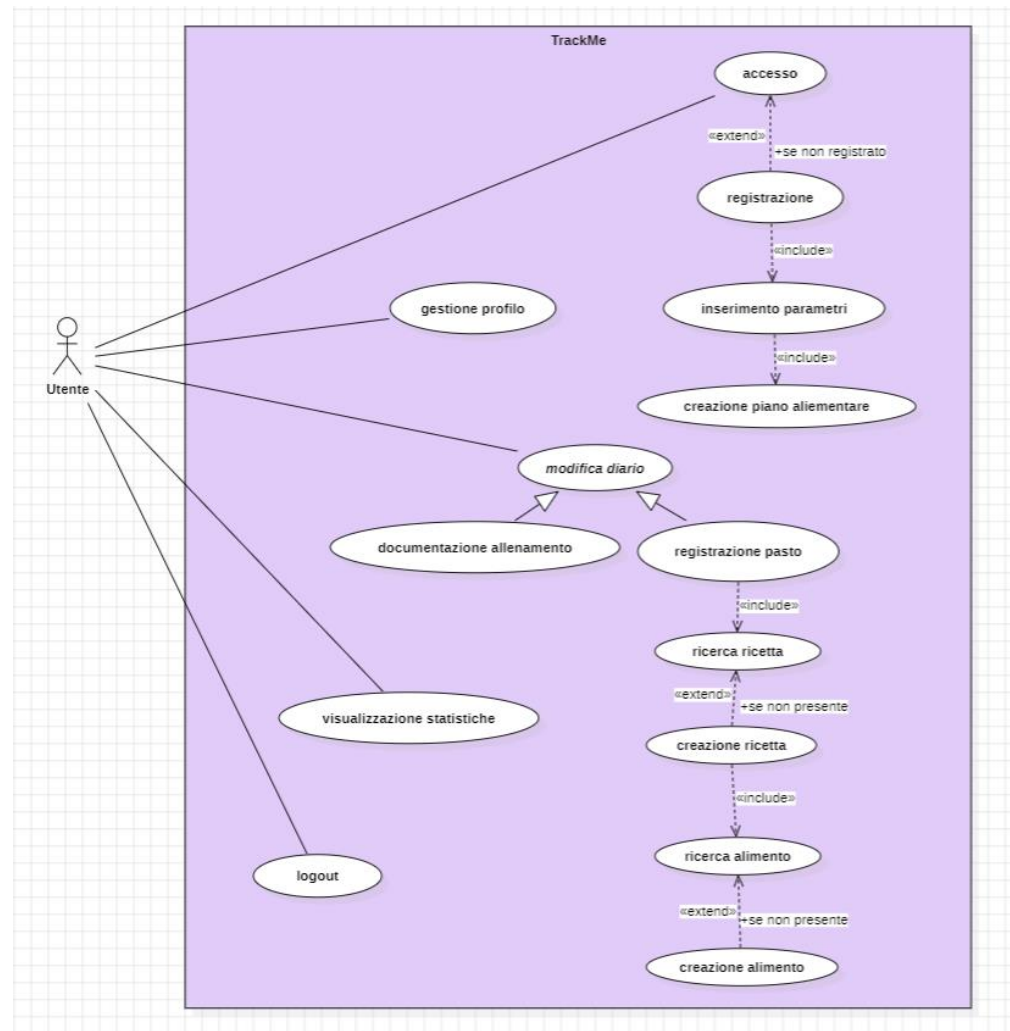




StarUML

Modellazione – diagrammi UML

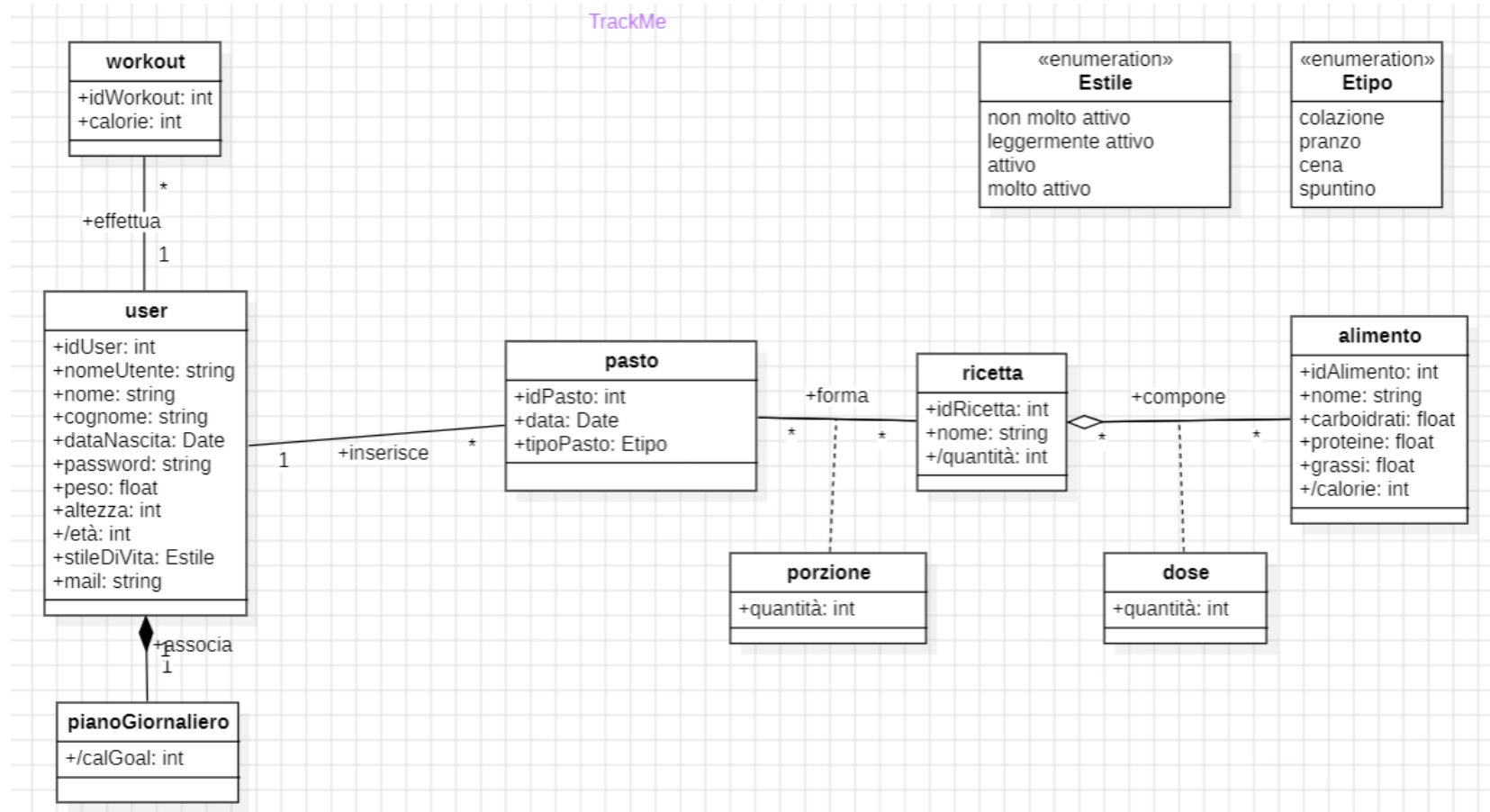
Use cases





Modellazione – diagrammi UML

Classes



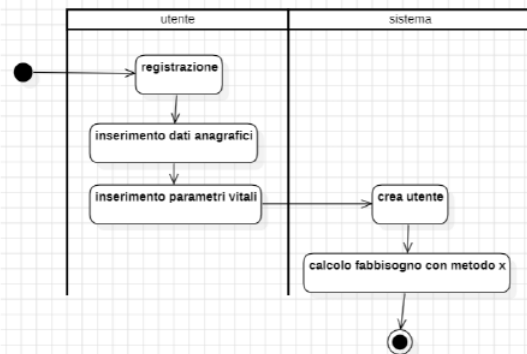


StarUML

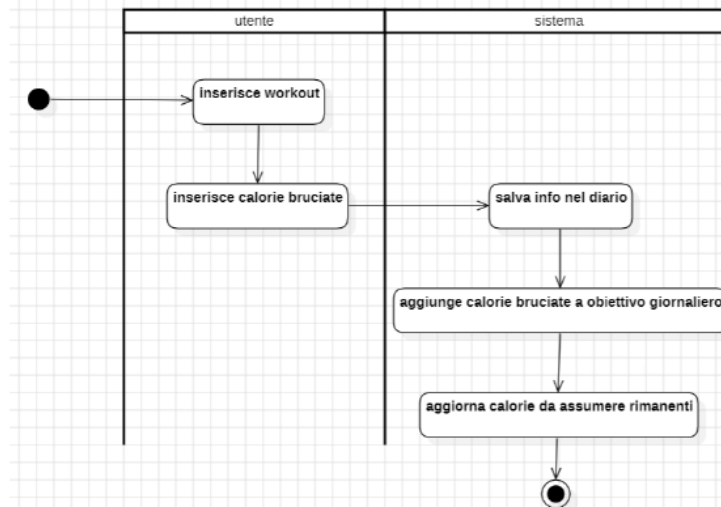
Modellazione – diagrammi UML

Activities

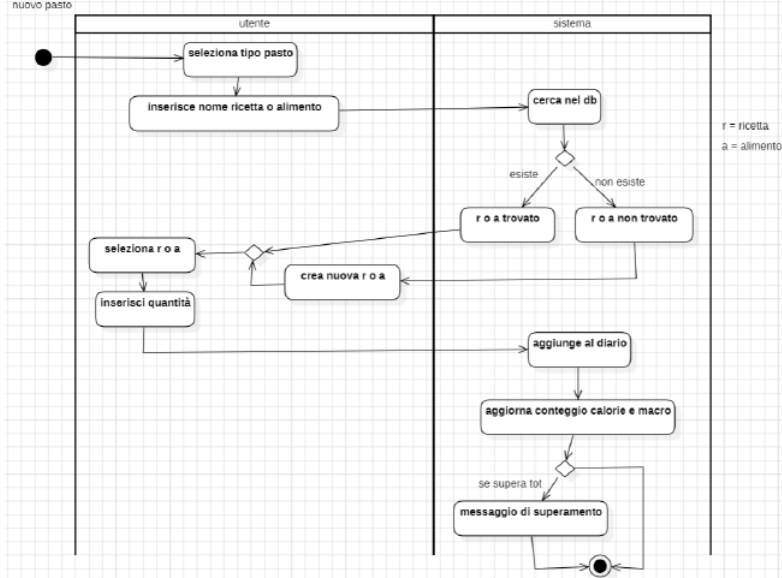
registrazione utente



registrazione workout

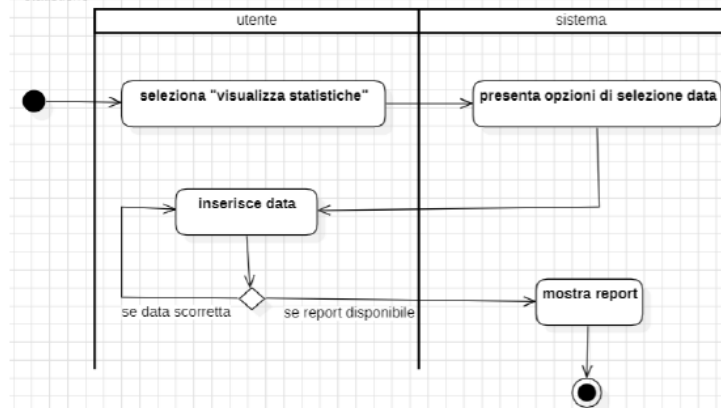


registrazione nuovo pasto



r = ricetta
a = alimento

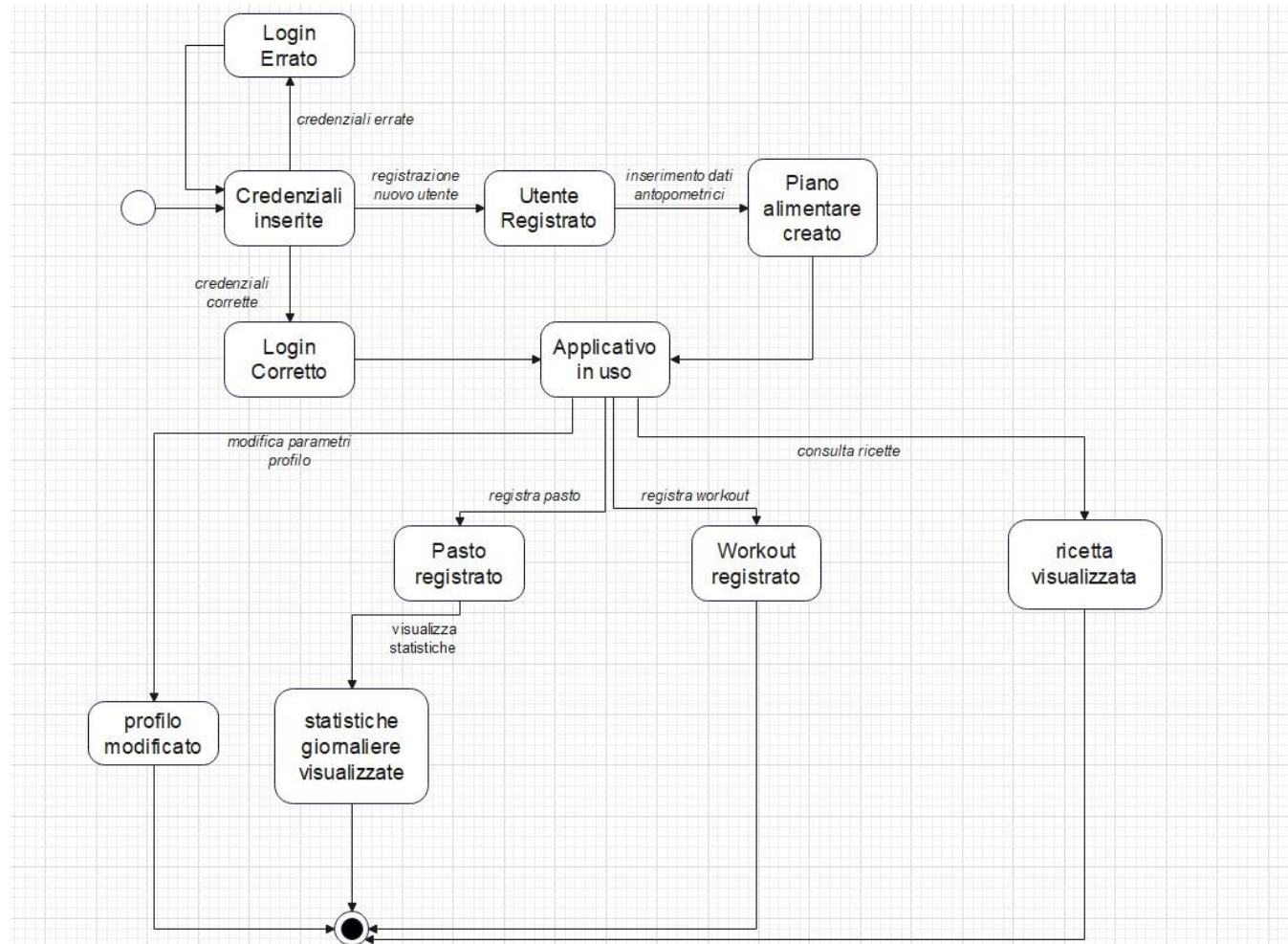
visualizza statistiche





Modellazione – diagrammi UML

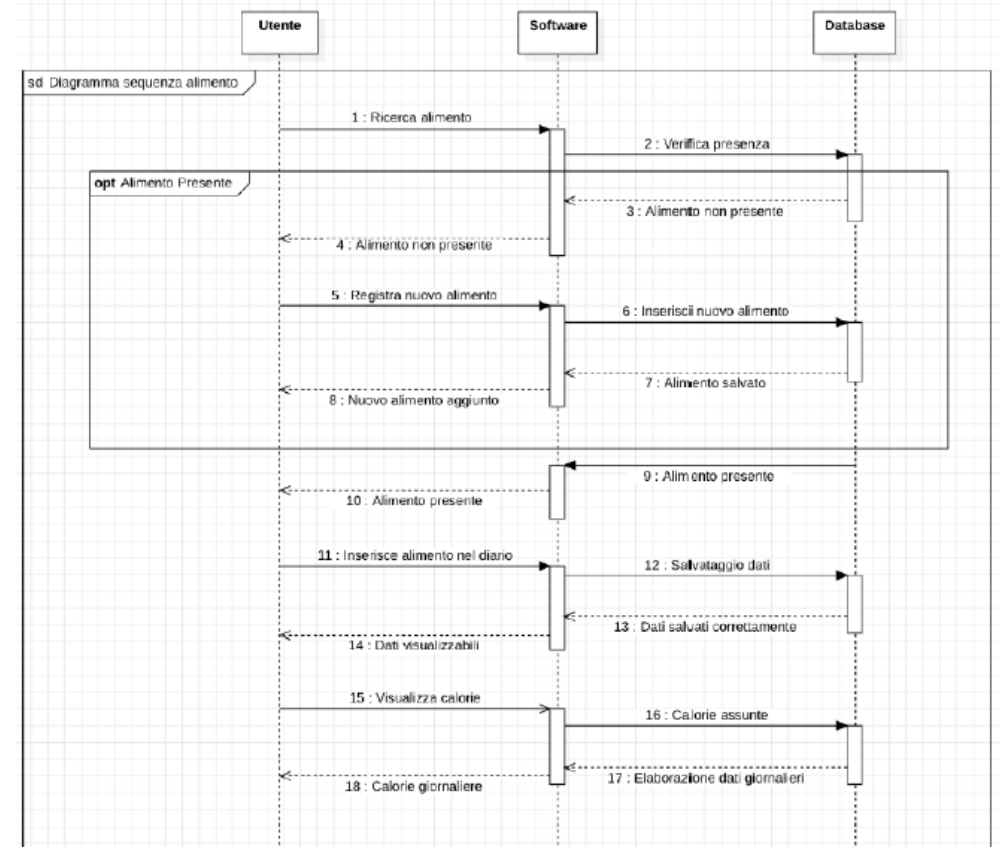
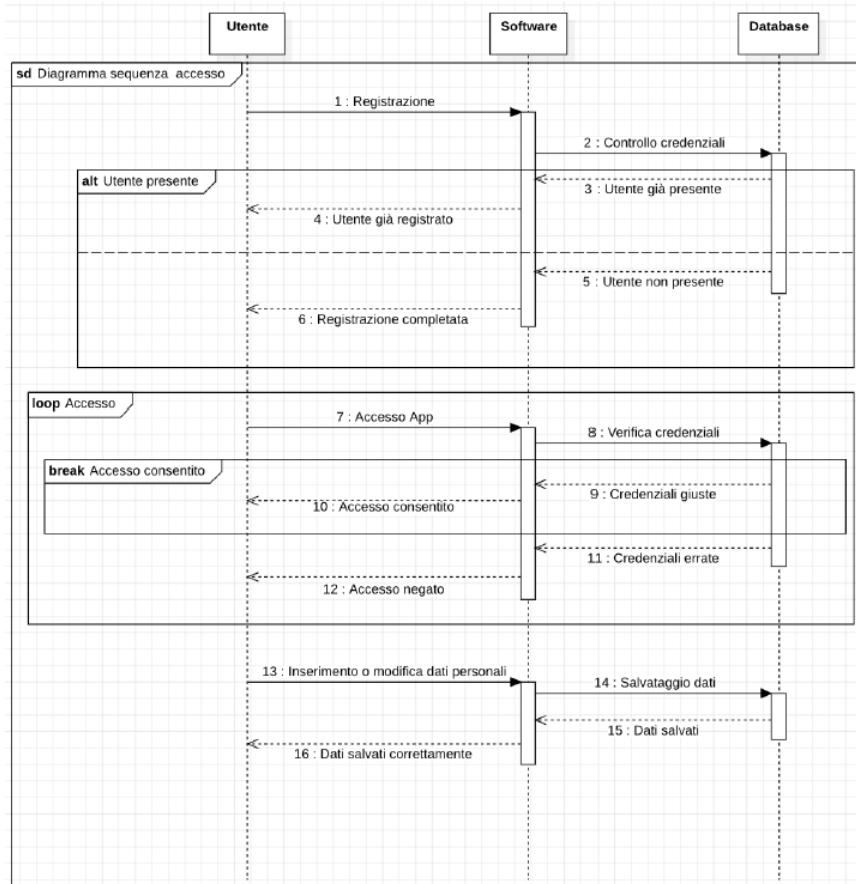
State





Modellazione – diagrammi UML

Sequence



Implementazione

Durante lo sviluppo del progetto, abbiamo implementato diverse funzionalità chiave:

- ✓ **Registrazione e accesso:** Gli utenti possono **registrarsi** per un nuovo account e **accedere** utilizzando le proprie credenziali.
- ✓ **Gestione del profilo:** Gli utenti possono **visualizzare** le informazioni del proprio profilo, inclusi obiettivi nutrizionali e preferenze personali.
- ✓ **Registrazione dei pasti:** Gli utenti possono **registrare** i pasti consumati durante la giornata, specificando dettagli come tipo di pasto e il sistema restituirà l'apporto calorico di ciò che viene inserito.
- ✓ **Creazione di ricette:** Gli utenti possono **creare** e **salvare** le proprie ricette personalizzate, specificando gli ingredienti e le quantità.
- ✓ **Monitoraggio delle calorie:** Il sistema tiene traccia **delle calorie consumate** dagli utenti in base ai pasti registrati e fornisce **statistiche** per il monitoraggio dell'apporto calorico giornaliero.
- ✓ **Gestione delle attività fisiche:** Gli utenti possono **registrare** le proprie attività fisiche e le calorie bruciate durante l'allenamento.
- ✓ **Interfaccia utente intuitiva:** L'applicazione offre un'interfaccia utente **intuitiva** e **facile** da usare, progettata per garantire un'**esperienza piacevole** e senza intoppi agli utenti.


The screenshot displays the TrackMe application interface. On the left is a blue sidebar with the TrackMe logo and navigation links: Stats, Recipes, and Settings. The main content area is light gray and contains several sections:

- Diary:** A section for tracking daily intake.
- Add a new Food:** A form with input fields for Food Name, Fat (100gr), Carbs (100gr), Protein (100gr), and Amount (set to 75). It includes an "Add food" button.
- Add a new Recipe:** A form with input fields for Recipe Name and Amount, and a text area for ingredients (containing "alette di pollo"). It includes an "Add Recipe" button.
- Add new meal:** A form with a large text area for meal description, a dropdown menu for Meal type (set to "pranzo"), and an input field for Amount. It includes an "Add Meal" button.
- Add workout:** A form with an input field for Calories. It includes an "Add Workout" button.

Implementazione

Abbiamo inoltre dovuto rimandare l'implementazione di alcune funzionalità, sulla base del modello MOSCOW, a causa delle tempistiche di consegna:

- ✓ **Consigli e suggerimenti:** Il sistema fornisce **consigli personalizzati** e **suggerimenti** per un'alimentazione sana e bilanciata in base alle preferenze e agli obiettivi dell'utente.
- ✓ **Calendario degli alimenti:** Una funzionalità per **visualizzare** e **pianificare** i pasti futuri degli utenti, consentendo loro di programmare e monitorare l'apporto calorico in anticipo.
- ✓ **Notifiche e promemoria:** Un sistema di notifiche per **ricordare** agli utenti di registrare i pasti, di **raggiungere** i loro **obiettivi** nutrizionali e di **mantenere** uno stile di vita sano.
- ✓ **Analisi avanzata dei dati:** Un'**analisi approfondita** delle abitudini alimentari degli utenti, con grafici dettagliati e statistiche per identificare tendenze e fornire raccomandazioni personalizzate.
- ✓ **Social sharing:** La possibilità per gli utenti di **condividere** i propri progressi, ricette e obiettivi con la comunità, incoraggiando l'**interazione** e il **supporto** tra gli utenti.



Settings menu

Name:	<input type="text" value="filippo"/>
Surname:	<input type="text" value="drago"/>
Date of Birth:	<input type="text" value="2024-02-06"/>
Username:	<input type="text" value="dragone64"/>
Daily goal:	<input type="text" value="1358"/>

[Back to Diary](#)

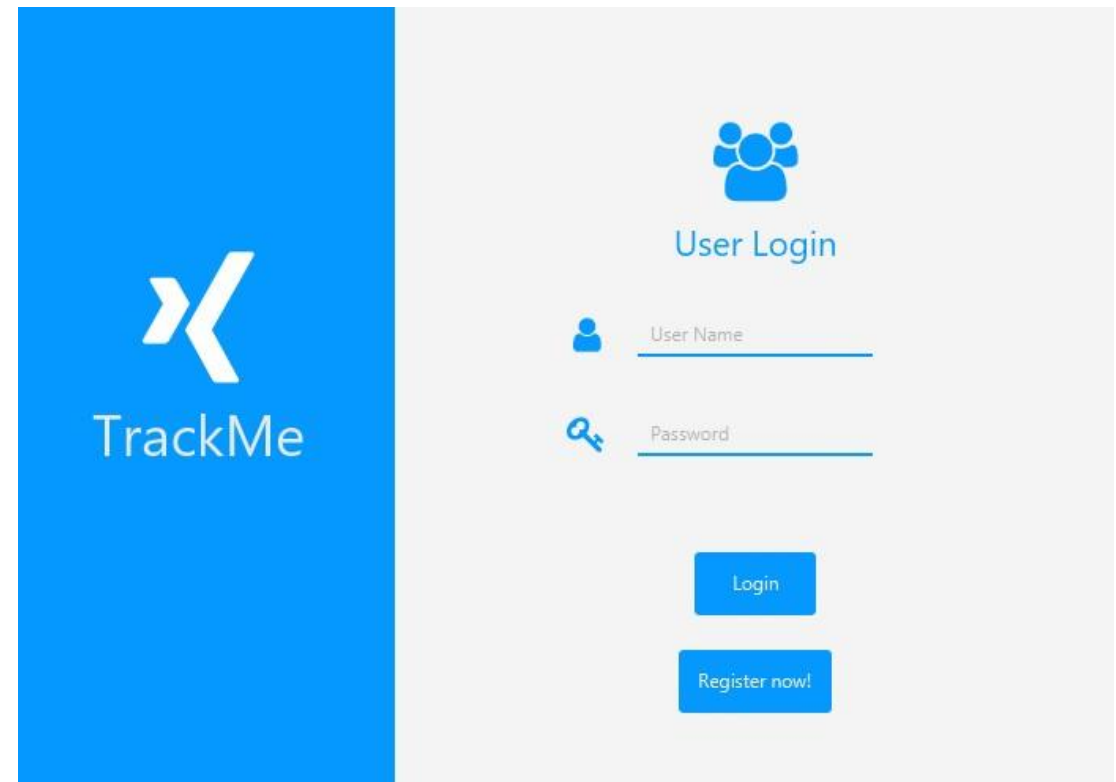
Queste funzionalità rimandate rappresentano potenziali aggiunte future al progetto, una volta che le risorse e le tempistiche lo permetteranno.

DEMO: Applicazione per monitoraggio calorie

Obiettivo: mostrare le **funzionalità chiave** dell'applicazione e l'esperienza dell'utente nell'utilizzarla per monitorare l'apporto calorico quotidiano e raggiungere gli obiettivi di salute e benessere.

Contenuti:

- **Registrazione e accesso:** Dimostrazione del processo di **registrazione** e **accesso** dell'utente all'applicazione.
- **Dashboard principale:** Visualizzazione della **schermata principale** dell'applicazione in cui l'utente può visualizzare gli alimenti del pasto consumato e **creare ricette** personalizzate.
- **Registrazione dei pasti:** Dimostrazione del processo per **registrare** e **monitorare** i pasti consumati dall'utente, inclusa la ricerca di alimenti e la registrazione delle porzioni.
- **Impostazioni del profilo:** Visualizzare le opzioni del **profilo** utente, inclusi gli obiettivi calorici.



Benefici della Demo

- **Illustra l'utilità e la facilità d'uso dell'applicazione per gli utenti.**
- **Mostra le caratteristiche distintive e i punti di forza dell'applicazione.**
- **Incoraggia l'interesse e l'adozione dell'applicazione.**

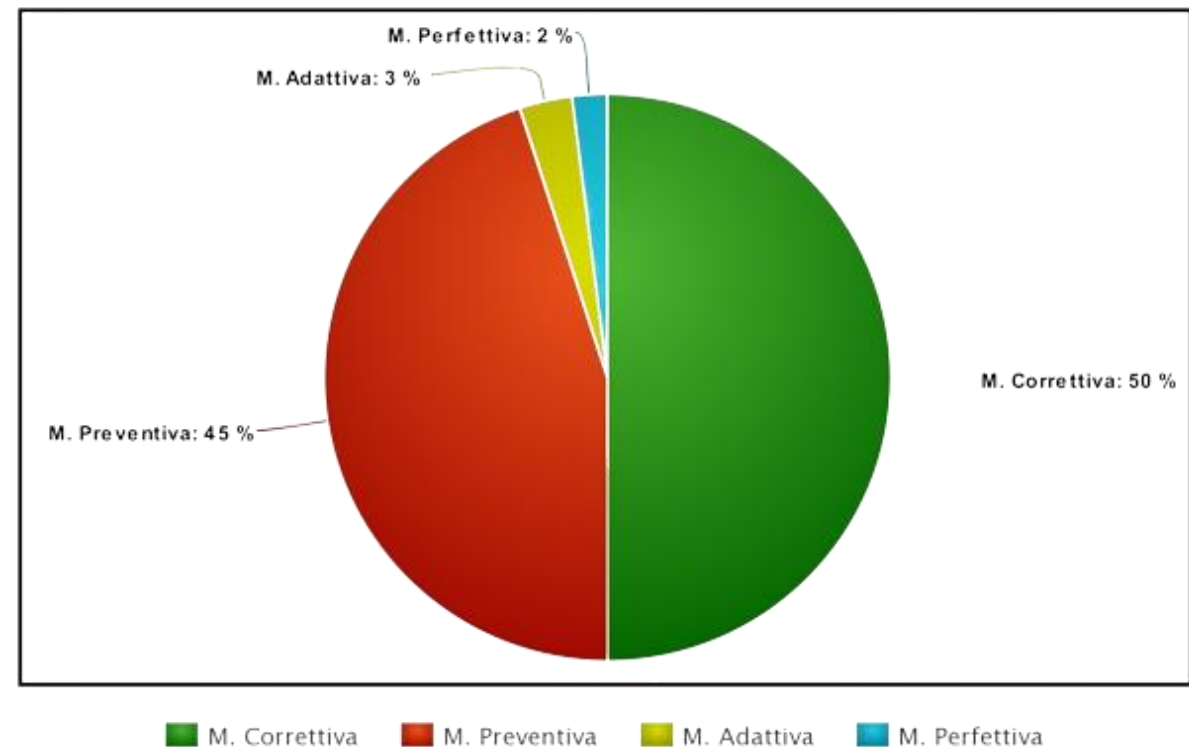
Manutenzione

Importanza

- Garantire il **corretto funzionamento** dell'applicazione nel tempo.
- **Migliorare** continuamente le **funzionalità** esistenti e **introdurre** nuove caratteristiche.
- **Risolvere** eventuali **problemi** o **bug** che possono emergere durante l'uso dell'applicazione.

Tipologie

- **Correttiva:** **Risoluzione** di **bug** e **problemi** identificati dagli utenti o durante i test.
- **Adattativa:** **Adattamento** dell'applicazione a **cambiamenti** nell'**ambiente operativo**, come aggiornamenti di sistema o nuovi requisiti di sicurezza.
- **Evolutiva:** Introduzione di **nuove funzionalità** e **miglioramenti** per rispondere alle esigenze degli utenti e alle tendenze del mercato.
- **Preventiva:** Attività volte a **prevenire** problemi futuri, come ottimizzazioni delle prestazioni e aggiornamenti regolari.



Manutenzione

Processo

- **Identificazione del problema:** Rilevamento dei bug o delle aree di miglioramento tramite segnalazioni, monitoraggio delle prestazioni e test regolari.
- **Analisi e risoluzione:** Analisi approfondita del problema, sviluppo di soluzioni e implementazione delle correzioni o degli aggiornamenti necessari.
- **Test e verifica:** Verifica dell'efficacia delle correzioni o dei miglioramenti attraverso test.
- **Rilascio e monitoraggio:** Distribuzione delle correzioni o dei miglioramenti tramite aggiornamenti dell'applicazione e monitoraggio continuo delle prestazioni per assicurare il corretto funzionamento.

Ruolo del team

- Collaborazione nell'identificazione e nella risoluzione dei problemi.
- Sviluppo di nuove funzionalità e miglioramenti in base alle esigenze degli utenti.
- Aggiornamento della documentazione e del codice sorgente per garantire la comprensibilità e la facilità di manutenzione futura.



Testing

Il testing è un passaggio cruciale nello sviluppo del software. Attraverso una serie di test mirati, verifichiamo il **corretto funzionamento** dell'applicazione e garantiamo una migliore esperienza utente. **Scopriamo e correggiamo** eventuali **difetti** prima del rilascio, assicurando la **qualità** e l'**affidabilità** del prodotto finale.

- **GestisciAlimentoTest:** Verifica se l'applicazione gestisce correttamente l'inserimento di un alimento già presente nel database, fornendo un errore appropriato.
- **GestisciComposizioneRicettaTest:** Valuta la corretta generazione di una ricetta e il collegamento con più alimenti nel sistema.
- **GestisciCreazionePastoTest:** Verifica se il processo di creazione di un pasto è correttamente collegato all'utente e alla ricetta selezionata.
- **GestisciPastoTest:** Controlla se l'applicazione genera correttamente un pasto, includendo tutti i dettagli necessari.
- **GestionePianoAlimentareTest:** Testa la generazione accurata delle calorie medie giornaliere e/o per un periodo specificato.
- **GestisciUtenteTest:** Verifica se il sistema gestisce correttamente l'inserimento di un utente già registrato o di una data successiva a quella odierna, generando gli errori appropriati. Si assicura anche che l'inserimento di un nuovo utente nel sistema sia gestito correttamente.





Grazie per
l'attenzione

- Aceti Alessia (1079023)
- De Vivo Massimo (1081227)
- Simoni Nicole (1080570)