

Relazione su "Daidokoro"
Elaborato per il corso di Basi di Dati

Matteo Giorgini - 0001136576
Tommaso De Tommaso - 0001077338
Edoardo Scorza - 0001077424

21 luglio 2024

Indice

1	Analisi dei requisiti	3
1.1	Requisiti in linguaggio naturale	3
1.2	Analisi ed estrazione dei concetti	4
1.2.1	Entità individuate	4
1.2.2	Relazioni individuate	5
2	Progettazione concettuale	6
2.1	Schema scheletro	6
2.2	Raffinamenti proposti	7
2.2.1	Dieta	7
2.2.2	Collezione	7
2.2.3	Obiettivo	7
2.2.4	Valutazione	7
2.3	Schema Concettuale	8
2.3.1	Valutazione	8
2.3.2	Obiettivo	9
2.3.3	Categoria	9
2.3.4	Collezione	10
3	Progettazione logica	11
3.1	Volume dei dati	12
3.2	Operazioni principali, Accessi e frequenza	13
3.3	Raffinamento schema	13
3.3.1	Eliminazione delle gerarchie	13
3.3.2	Eliminazione degli attributi composti	14
3.3.3	Scelta delle chiavi primarie	14
3.3.4	Chiavi Esterne	15
3.4	Traduzione di entità e relazioni in associazioni	16
3.5	Traduzione delle operazioni in query SQL	17
3.5.1	Query di creazione	17

3.5.2	Query di selezione	20
4	Progettazione applicazione	28
4.1	Descrizione dell'architettura dell'applicazione realizzata	28
4.2	Elenco delle operazioni	38
4.2.1	Operazioni di selezione	38
4.2.2	Operazioni di inserimento, aggiornamento e rimozione	40

Capitolo 1

Analisi dei requisiti

1.1 Requisiti in linguaggio naturale

Si vuole realizzare un database a supporto di un social network di ricette, questo dovrà tenere traccia degli *utenti* registrati salvandone il loro username, la foto profilo, l'email, la password ed un codice univoco per identificarli. Inoltre, si vuole anche tenere traccia dell'esperienza e del livello, i quali rappresentano il progresso di un utente nello sblocco di *obiettivi* che sono caratterizzati da un nome, una descrizione e dalla quantità di esperienza da dare all'utente una volta sbloccato. Questi vengono usati per incitare l'utente ad utilizzare tutte le diverse funzionalità del software, che possono anche essere limitate fino allo sblocco di un determinato obiettivo. L'utente, dopo aver effettuato l'accesso, potrà osservare le ricette altrui, aggiungerle ai preferiti, salvarle in collezioni o in diete e valutarle con un voto ed un commento. L'utente potrà anche pubblicare le proprie *ricette* inserendone il nome, la foto, la descrizione, la difficoltà e il tempo di realizzazione indicativo, oltre agli ingredienti che la compongono ed ai passi necessari a prepararla. Gli *ingredienti* sono composti da un nome, una descrizione, dai valori nutrizionali e dalle categorie nutrizionali a cui appartengono. Le *collezioni* possiedono un nome, una descrizione, la data di creazione e la lista di ricette che ne fanno parte. Ci sono anche le diete, che funzionano come le collezioni, ma differiscono dal fatto che devono aderire ad una *categoria nutrizionale* specifica, vincolando quindi le ricette che si possono aggiungere. Infine, le *valutazioni* che gli utenti possono lasciare a ricette, collezioni o diete, consistono in un voto positivo o negativo, eventualmente accompagnato da un commento se l'utente ha sbloccato l'apposito obiettivo.

1.2 Analisi ed estrazione dei concetti

Il testo dei requisiti fornisce una accurata descrizione delle entità chiave, tuttavia è anche abbastanza ambiguo sulla gestione dei vincoli generati dagli obiettivi, su dove memorizzare i valori nutrizionali degli ingredienti ed sui vincoli delle diete che accettano solo ricette con ingredienti di una certa categoria nutrizionale.

Molti dei vincoli funzionali che dipendono dallo sblocco di un obiettivo non sono modellabili nel database e quindi saranno implementati nella programmazione, tuttavia per differenziare tra la valutazione senza commento e quella con il commento, si modifica l'entità valutazione spostando il commento in un suo subset denominato ***critica***. I ***valori nutrizionali*** saranno memorizzati in una tabella separata che avrà una relazione 1 a 1 con la tabella degli ingredienti in modo da separare meglio dati non essenziali che non identificano direttamente l'ingrediente. Infine, le ***diete*** sono un particolare tipo di collezione e quindi vengono trasformate in un subset.

1.2.1 Entità individuate

Nome	Tipologia	Descrizione
Utente	Entità	Rappresenta gli utenti registrati al social network
Obiettivo	Entità	Rappresenta gli obiettivi sbloccabili dagli utenti
Ricetta	Entità	Rappresenta le ricette pubblicate dagli utenti
Ingrediente	Entità	Rappresenta gli ingredienti utilizzabili nelle ricette
Valore nutrizionale	Entità debole	Rappresenta i valori nutrizionali degli ingredienti
Collezione	Entità	Rappresenta le collezioni di ricette create dagli utenti
Dieta	Entità <i>Subset di Collezione</i>	Rappresenta le collezioni vincolate dalla categoria nutrizionale
Valutazione	Entità	Rappresenta le valutazioni degli utenti alle ricette, collezioni e diete
Critica	Entità <i>Subset di Valutazione</i>	Rappresenta le valutazioni che possiedono anche un commento
Categoria nutrizionale	Entità	Rappresenta le categorie nutrizionali utilizzati negli ingredienti, nelle diete e negli obiettivi

1.2.2 Relazioni individuate

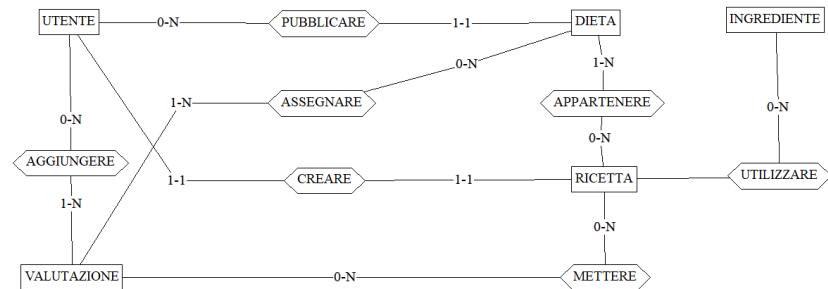
Nome	Tipologia	Descrizione
Aggiungere	Relazione <i>Utente - Valutazione</i>	Rappresenta la creazione di una valutazione da parte di un utente
Pubblicare	Relazione <i>Utente - Dieta</i>	Rappresenta la creazione di una dieta da parte di un utente
Assegnare	Relazione <i>Valutazione - Dieta</i>	Rappresenta il collegamento tra una valutazione e la dieta a cui si riferisce
Creare	Relazione <i>Utente - Ricetta</i>	Rappresenta la creazione di una ricetta da parte di un utente
Appartenere	Relazione <i>Ricetta - Dieta</i>	Rappresenta l'appartenenza di una ricetta ad una specifica dieta
Valutare	Relazione <i>Valutazione - Ricetta</i>	Rappresenta il collegamento tra una valutazione e la ricetta che valuta
Utilizzare	Relazione <i>Ricetta - Ingrediente</i>	Rappresenta l'utilizzo di un ingrediente in una ricetta
Sbloccare	Relazione <i>Utente - Obiettivo</i>	Rappresenta gli obiettivi sbloccati da un utente
Dipendere	Relazione <i>Obiettivo - CATEGORIA NUTRIZIONALE</i>	Rappresenta la dipendenza tra l'obiettivo e la categoria nutrizionale
Vincolare	Relazione <i>CATEGORIA NUTRIZIONALE - DIETA</i>	Rappresenta il vincolo della dieta di poter inserire solo ricette con ingredienti compatibili
Descrivere	Relazione <i>CATEGORIA NUTRIZIONALE - INGREDIENTE</i>	Rappresenta la caratterizzazione di un ingrediente con delle categorie nutrizionali

Capitolo 2

Progettazione concettuale

Per la progettazione concettuale abbiamo considerato diverse cose, tra cui, la organizzazione della applicazione e le capacità degli utenti, in primis, la app è di tipo client-server, quindi il database è unico e possiede tutti i dati del "ecosistema" (client e server), gli utenti sono potenzialmente tanti e hanno un frequente accesso al database, motivo per il cui i dati sono frammentati, separati in varie parti, per evitare un accesso non essenziale a grandi quantità di dati, e ridurre il carico sul database.

2.1 Schema scheletro



La prima versione dello schema evidenzia gli elementi chiave che costruiscono le informazioni su cui si basa la applicazione:

- Utente
- Ricetta

- Valutazione
- Ingrediente
- Dieta

2.2 Raffinamenti proposti

Analizzando meglio la possibile esperienza dell’utente sono molte le possibili aggiunte, abbiamo cercato dunque di rendere il database completo senza complicarlo eccessivamente:

2.2.1 Dieta

La dieta, corrisponde alla possibilità di raggruppare ricette per rendere più significativo il significato di essa abbiamo deciso di imporre dei limiti, come la categoria, che limita la tipologia di ricette inseribili, la creazione di essa è permessa all’utente solo dopo aver sbloccato un determinato obiettivo.

2.2.2 Collezione

Per non limitare le possibilità dell’utente abbiamo deciso di creare una variante più generale della dieta, la collezione essa permette di salvare ricette indiscriminatamente.

2.2.3 Obiettivo

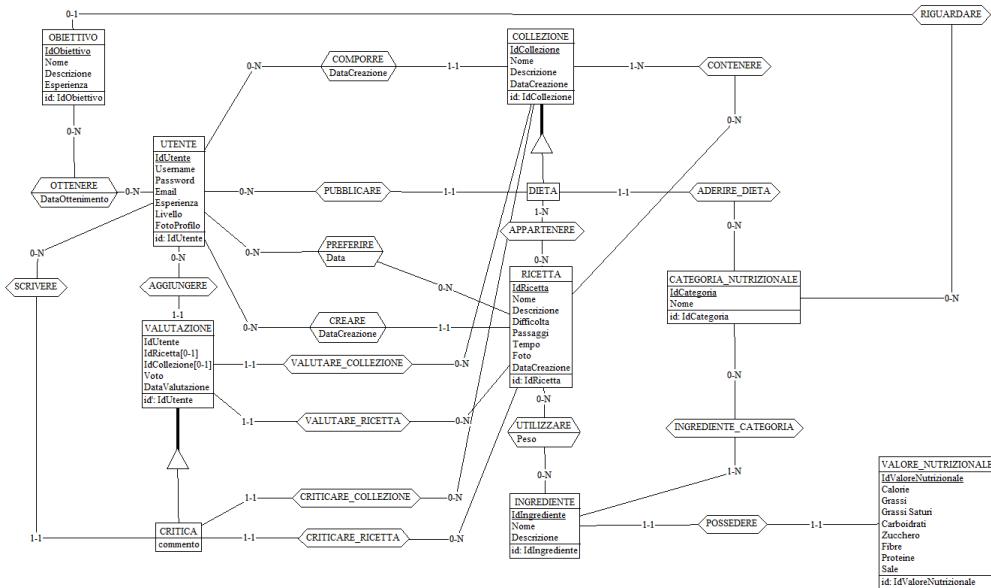
Per dare più autorevolezza agli utenti esperti e filtrare utenti ”troll” abbiamo deciso di implementare dei requisiti, per fare delle diete bisogna aver sbloccato un obiettivo, così per le critiche

2.2.4 Valutazione

Per le valutazioni abbiamo optato su 2 tipologie valutazione e critica, la prima consiste in un voto numerico, assegnabile da un utente generico, il secondo è una critica che aggiunge il commento, questa limitata da un obiettivo.

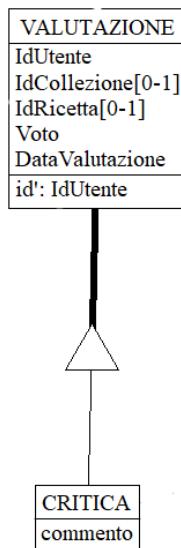
2.3 Schema Concettuale

Il risultato finale è questo schema:



2.3.1 Valutazione

Essendo critica una forma avanzata di valutazione abbiamo optato per una Gerarchia



2.3.2 Obiettivo

La necessità di imporre vincoli all'utente precedentemente espressa è implementata attraverso l'uso di obiettivi

OBIETTIVO
<u>IdObiettivo</u>
Nome
Descrizione
Esperienza
id: IdObiettivo

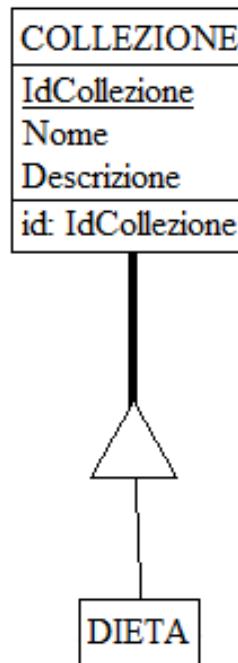
2.3.3 Categoria

Un altro limite atto a rendere la ricerca di ricette più mirata e le diete più consistenti è la categoria nutrizionale che si relaziona con gli ingredienti da cui indirettamente si ricavano le categorie per una ricetta.

CATEGORIA_NUTRIZIONALE
<u>IdCategoria</u>
Nome
id: IdCategoria

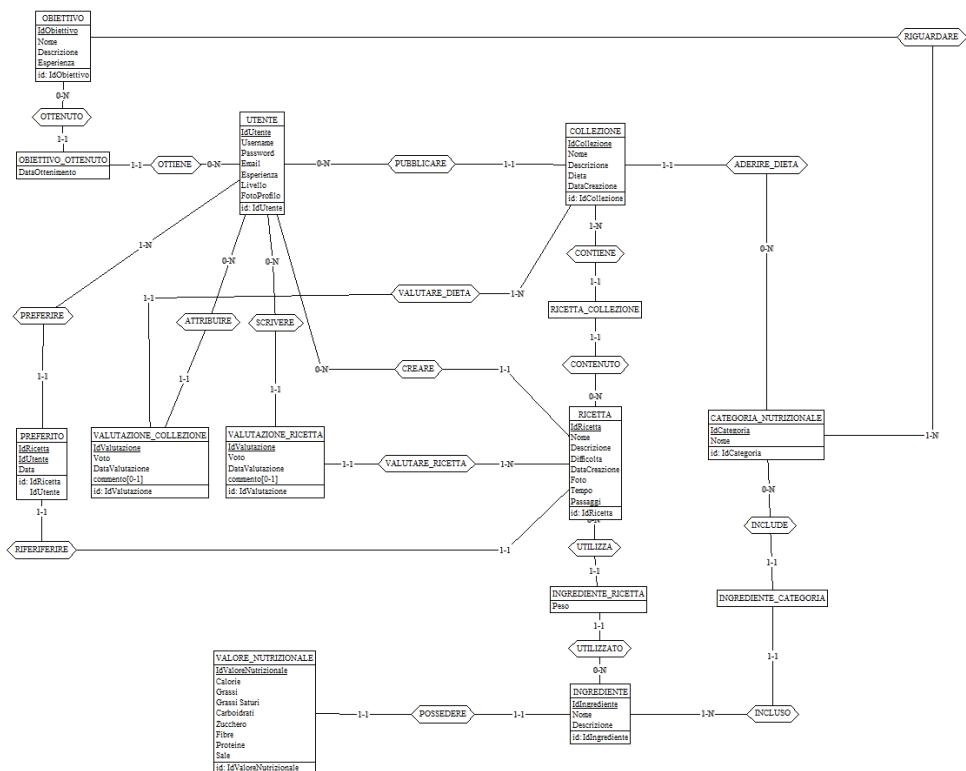
2.3.4 Collezione

Molto similmente alle valutazioni, per ottenere una versione limitata ma libera a tutti gli utenti e una versione di collezione completa(la dieta) ma non disponibile immediatamente, abbiamo optato per una gerarchia.



Capitolo 3

Progettazione logica



3.1 Volume dei dati

Concetto	Costrutto	Volume
Utente	E	10'000
Aggiungere	R	1'000'000
Valutazione	E	1'000'000
Ottenerе	R	1'000'000
Obiettivo	E	20
Publicare	R	10'000'000
Dieta	E	1'000'000
Ricetta	E	10'000'000
Valutare_ricetta	R	30'000'000
Valutare_dieta	R	7'000'000
Preferire	R	30'000'000
Appartenere	R	10'000'000
Utilizzare	R	60'000'000
Ingrediente	E	3'000
Possedere	R	3'000
Valore_Nutrizionale	E	3'000
Comporre	R	100'000
Collezione	E	100'000
Scrivere	R	200'000
Critica	E	200'000
Criticare_Dieta	R	80'000
Criticare_Ricetta	R	300'000
Riguardare	R	20
Categoria_nutrizionale	E	20
Aderire_dieta	R	20
Comporre	R	100'000
Contenere	R	1'000'000

3.2 Operazioni principali, Accessi e frequenza

Operazione	Frequenza-giornaliera	Accessi per operazione	Accessi-giornalieri
Pubblicare ricetta	20'000	1S	40'000
Aggiungere valutazione	100'000	1S	200'000
Ricerca-Ricetta	1'000'000	4L	4'000'000
Ricerca-Dieta	10'000	5L	50'000
Inserimento Utente	10	1S	60
Ottenere Obiettivo	30	1S	60
Vedere Tendenze	3'000	4L	12'000
Pubblicare Collezione	7'000	1S	14'000
Visualizzare recensioni	700	2L	1400
Visualizzare obiettivi	1500	2L	3000
Visualizzare preferiti	30'000	2L	60'000

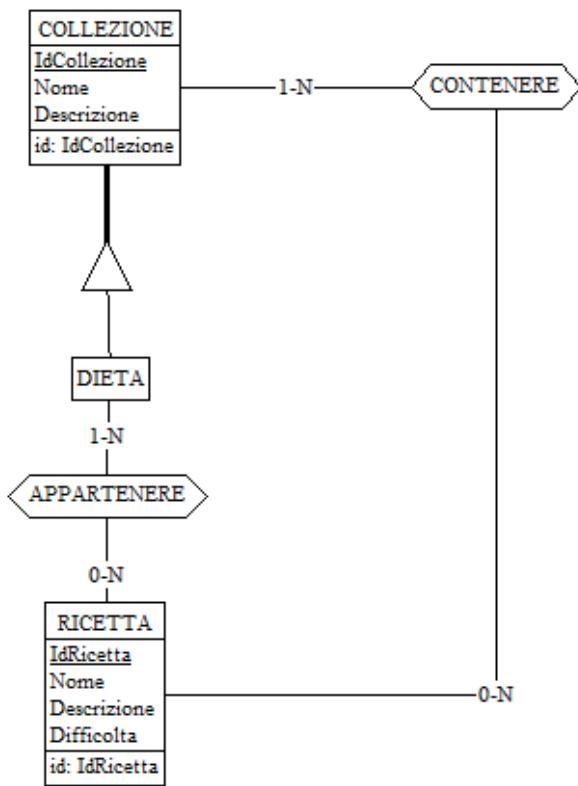
3.3 Raffinamento schema

3.3.1 Eliminazione delle gerarchie

Nello schema E/R sono presenti 2 gerarchie da eliminare.

Per la gerarchia COLLEZIONE(t, e), si decide di adottare come soluzione il collasso verso l'alto. Abbiamo, quindi, aggiunto un attributo, di tipo booleano, Dieta che indica se si tratta di una semplice collezione o di una dieta. La scelta è motivata dal fatto che le associazioni APPARTENERE e CONTENERE tra COLLEZIONE/DIETA e RICETTA sono del tipo 1-N, quindi si preferisce avere un'unica tabella che contiene tutte le collezioni piuttosto che due tabelle divise che non avrebbero alcuna utilità, poiché gli accessi all'entità padre e alle entità figlie sono contestuali.

Anche per la gerarchia VALUTAZIONE(t, e) si adotta una soluzione simile, viene aggiunto l'attributo Commento che rende la valutazione una critica quando non è vuoto, mentre per la questione delle due chiavi opzionali, in quanto abbiamo bisogno di una chiave o l'altra, abbiamo preferito separare la nuova entità valutazione/critica in 2 entità valutazione-collezione e



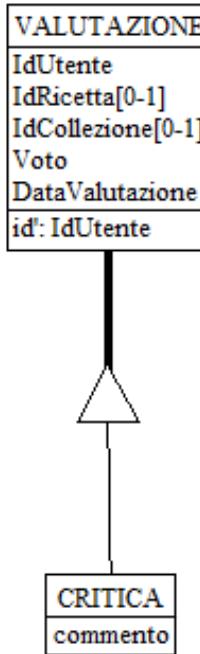
valutazione-ricetta

3.3.2 Eliminazione degli attributi composti

Nel nostro schema non sono presenti attributi composti o multipli per nostra scelta, tuttavia se volessimo individuare un possibile caso, quello sarebbe valore nutrizionale come campo multiplo di ingrediente.

3.3.3 Scelta delle chiavi primarie

Data la natura del nostro dominio, non presentando innatamente un forte sistema di identificazione Ogni entità possiederà una chiave unica.



3.3.4 Chiavi Esterne

Le relazioni sono state risolte in base alla loro cardinalità rispetto alle entità connesse, non essendoci casi particolari, le relazioni si sono risolte importando una chiave esterna nella rispettiva entità , oppure creando una tabella dedicata, nel caso di relazioni N-N:

- **Obiettivo ottenuto da utente**, 3 Entità 2 Relazioni, Vengono importate le chiavi di utente e obiettivo in obiettivo-ottenuto
- **Utente pubblica collezione**, 2 Entità 1 Relazione, viene importata la chiave di utente dentro collezione
- **Collezione possiede ricetta** 3 Entità 2 Relazioni, In ricetta-collezione vengono salvate le chiavi di Collezione e ricetta
- **Ricetta possiede ingrediente** 3 Entità 2 Relazioni, in ingrediente-ricetta vengono importate le chiavi di ingrediente e ricetta
- **Ingrediente aderisce a categoria nutrizionale** 3 Entità 2 Relazioni, viene importato in ingrediente-categoria le chiavi di ingrediente e categoria
- **Utente Valuta collezione** 3 Entità 2 Relazioni,dentro valutazione-collezione vengono salvate le chiavi di valutazione e collezione

3.4 Traduzione di entità e relazioni in associazioni

UTENTE(IdUtente, Username, Foto, Pwd, Email, Esperienza, Livello)

CATEGORIA_NUTRIZIONALE(IdCategoria, Nome)

OBIETTIVO(IdObiettivo, Nome, Descrizione, Esperienza, IdCategoria)
FK: IdCategoria REFERENCES **CATEGORIA_NUTRIZIONALE**

OBIETTIVO_OTTENUTO(IdObiettivo, IdUtente, DataCreazione)
FK: IdObiettivo REFERENCES **OBIETTIVO**
FK: IdUtente REFERENCES **UTENTE**

RICETTA(IdRicetta, Nome, Descrizione, Passaggi, Foto, Difficoltà, Tempo, DataCreazione, IdUtente)
FK: IdUtente REFERENCES **UTENTE**

COLLEZIONE(IdCollezione, Nome, Descrizione, Dieta, DataCreazione, IdUtente, IdCategoria)
FK: IdUtente REFERENCES **UTENTE**
FK: IdCategoria REFERENCES **CATEGORIA_NUTRIZIONALE**

INGREDIENTE(IdIngrediente, Nome, Descrizione, IdValoreNutrizionale, IdCategoria)
FK: IdCategoria REFERENCES **CATEGORIA_NUTRIZIONALE**
FK: IdValoreNutrizionale REFERENCES **VALORE_NUTRIZIONALE**

VALORE_NUTRIZIONALE(IdValoreNutrizionale, Calorie, Grassi, Grassi_Saturi, Carboidrati, Zucchero, Fibre, Proteine, Sale, IdIngrediente)
FK: IdIngrediente REFERENCES **INGREDIENTE**

RICETTA_COLLEZIONE(IdRicetta, IdCollezione)
FK: IdRicetta REFERENCES **RICETTA**
FK: IdCollezione REFERENCES **COLLEZIONE**

INGREDIENTE_RICETTA(IdIngrediente, IdRicetta, PesoInGrammi)
FK: IdIngrediente REFERENCES **INGREDIENTE**
FK: IdRicetta REFERENCES **RICETTA**

LIKES(IdUtente, IdRicetta, Data)

FK: IdUtente REFERENCES UTENTE

FK: IdRicetta REFERENCES RICETTA

VALUTAZIONE_RICETTA(IdRicetta, IdUtente, Voto,
DataValutazione, Commento)

FK: IdUtente REFERENCES UTENTE

FK: IdRicetta REFERENCES RICETTA

VALUTAZIONE_COLLEZIONE(IdCollezione, IdUtente, Voto,
DataValutazione, Commento)

FK: IdUtente REFERENCES UTENTE

FK: IdCollezione REFERENCES COLLEZIONE

3.5 Traduzione delle operazioni in query SQL

3.5.1 Query di creazione

```
CREATE DATABASE IF NOT EXISTS daidokoro;
USE daidokoro;

CREATE TABLE IF NOT EXISTS utente(
    IdUtente INT NOT NULL AUTO_INCREMENT,
    Username VARCHAR(20) NOT NULL,
    Foto MEDIUMBLOB NOT NULL,
    Pwd VARCHAR(30) NOT NULL,
    Email VARCHAR(30) NOT NULL,
    Esperienza INT DEFAULT 0,
    Livello INT DEFAULT 1,
    PRIMARY KEY (IdUtente)
)AUTO_INCREMENT=1;

CREATE TABLE IF NOT EXISTS categoria_nutrizionale(
    IdCategoria INT NOT NULL AUTO_INCREMENT,
    Nome VARCHAR(30) NOT NULL,
    PRIMARY KEY (IdCategoria)
)AUTO_INCREMENT=1;

CREATE TABLE IF NOT EXISTS obiettivo(
    IdObiettivo INT NOT NULL AUTO_INCREMENT,
    Nome VARCHAR(30) NOT NULL,
    Descrizione VARCHAR(255) NOT NULL,
    Esperienza INT NOT NULL,
    IdCategoria INT NOT NULL,
    PRIMARY KEY (IdObiettivo),
    FOREIGN KEY (IdCategoria) REFERENCES categoria_nutrizionale(IdCategoria)
)AUTO_INCREMENT=1;

CREATE TABLE IF NOT EXISTS obiettivo_ottenuto(
```

```

IdObiettivo INT NOT NULL,
IdUtente INT NOT NULL,
DataOttenimento DATE NOT NULL,
FOREIGN KEY (IdObiettivo) REFERENCES obiettivo(IdObiettivo),
FOREIGN KEY (IdUtente) REFERENCES utente(IdUtente),
PRIMARY KEY (IdObiettivo, IdUtente)
);

CREATE TABLE IF NOT EXISTS valore_nutrizionale(
IdValoreNutrizionale INT NOT NULL AUTO_INCREMENT,
Calorie FLOAT NOT NULL,
Grassi FLOAT NOT NULL,
Grassi_Saturi FLOAT NOT NULL,
Carboidrati FLOAT NOT NULL,
Zucchero FLOAT NOT NULL,
Fibre FLOAT NOT NULL,
Proteine FLOAT NOT NULL,
Sale FLOAT NOT NULL,
IdIngrediente INT NOT NULL,
PRIMARY KEY (IdValoreNutrizionale)
)AUTO_INCREMENT=1;

CREATE TABLE IF NOT EXISTS ingrediente(
IdIngrediente INT NOT NULL AUTO_INCREMENT,
Nome VARCHAR(50) NOT NULL,
Descrizione VARCHAR(255) NOT NULL,
IdValoreNutrizionale INT NULL,
IdCategoria INT NOT NULL,
PRIMARY KEY (IdIngrediente),
FOREIGN KEY (IdCategoria) REFERENCES categoria_nutrizionale(IdCategoria)
)AUTO_INCREMENT=1;

CREATE TABLE IF NOT EXISTS ricetta(
IdRicetta INT NOT NULL AUTO_INCREMENT,
Nome VARCHAR(50) NOT NULL,
Descrizione VARCHAR(255) NOT NULL,
Passaggi JSON NOT NULL,
Foto MEDIUMBLOB NOT NULL,
Difficoltà INT NOT NULL,
Tempo INT NOT NULL,
DataCreazione DATE NOT NULL,
IdUtente INT NOT NULL,
PRIMARY KEY (IdRicetta),
FOREIGN KEY (IdUtente) REFERENCES utente(IdUtente)
)AUTO_INCREMENT=1;

CREATE TABLE IF NOT EXISTS likes (
IdUtente INT NOT NULL,
IdRicetta INT NOT NULL,
Data DATE NOT NULL,
FOREIGN KEY (IdUtente) REFERENCES utente(IdUtente),
FOREIGN KEY (IdRicetta) REFERENCES ricetta(IdRicetta),
PRIMARY KEY (IdUtente, IdRicetta)
);

CREATE TABLE IF NOT EXISTS ingrediente_ricetta(
IdIngrediente INT NOT NULL,
IdRicetta INT NOT NULL,
PesoInGrammi INT NOT NULL,
FOREIGN KEY (IdIngrediente) REFERENCES ingrediente(IdIngrediente),
FOREIGN KEY (IdRicetta) REFERENCES ricetta(IdRicetta),
PRIMARY KEY (IdIngrediente, IdRicetta)
);

```

```

);

CREATE TABLE IF NOT EXISTS collezione(
    IdCollezione INT NOT NULL AUTO_INCREMENT,
    Nome VARCHAR(50) NOT NULL,
    Descrizione VARCHAR(255) NOT NULL,
    Dieta TINYINT(1) NOT NULL,
    DataCreazione DATE NOT NULL,
    IdUtente INT NOT NULL,
    IdCategoria INT NOT NULL,
    PRIMARY KEY (IdCollezione),
    FOREIGN KEY (IdUtente) REFERENCES utente(IdUtente),
    FOREIGN KEY (IdCategoria) REFERENCES categoria_nutrizionale(IdCategoria)
)AUTO_INCREMENT=1;

CREATE TABLE IF NOT EXISTS ricetta_collezione(
    IdRicetta INT NOT NULL,
    IdCollezione INT NOT NULL,
    FOREIGN KEY (IdRicetta) REFERENCES ricetta(IdRicetta),
    FOREIGN KEY (IdCollezione) REFERENCES collezione(IdCollezione),
    PRIMARY KEY (IdRicetta, IdCollezione)
);

CREATE TABLE IF NOT EXISTS valutazione_ricetta(
    IdRicetta INT NOT NULL,
    IdUtente INT NOT NULL,
    Voto TINYINT(1) NOT NULL,
    DataValutazione DATETIME NOT NULL,
    Commento VARCHAR(255) DEFAULT '',
    PRIMARY KEY (IdUtente, IdRicetta),
    FOREIGN KEY (IdUtente) REFERENCES utente(IdUtente),
    FOREIGN KEY (IdRicetta) REFERENCES ricetta(IdRicetta)
);

CREATE TABLE IF NOT EXISTS valutazione_collezione(
    IdCollezione INT NOT NULL,
    IdUtente INT NOT NULL,
    Voto TINYINT(1) NOT NULL,
    DataValutazione DATETIME NOT NULL,
    Commento VARCHAR(255) DEFAULT '',
    PRIMARY KEY (IdUtente, IdCollezione),
    FOREIGN KEY (IdUtente) REFERENCES utente(IdUtente),
    FOREIGN KEY (IdCollezione) REFERENCES collezione(IdCollezione)
);

ALTER TABLE ingrediente
ADD FOREIGN KEY (IdValoreNutrizionale)
REFERENCES valore_nutrizionale(IdValoreNutrizionale);

ALTER TABLE valore_nutrizionale
ADD FOREIGN KEY (IdIngrediente)
REFERENCES ingrediente(IdIngrediente);

INSERT INTO categoria_nutrizionale (Nome)
VALUES ("Generico");

INSERT INTO obiettivo (Nome, Descrizione, Esperienza, IdCategoria)
VALUES ("Benvenuto", "Ti sei registrato a Daidokoro!", 100, 1);

```

3.5.2 Query di selezione

```
SELECT ingrediente.*, ingrediente_ricetta.PesoInGrammi AS Peso
FROM ingrediente JOIN ingrediente_ricetta
ON ingrediente.IdIngrediente = ingrediente_ricetta.IdIngrediente
WHERE ingrediente_ricetta.IdRicetta = ?;

WITH v1(IdRicetta, Nome, Descrizione, Passaggi, Foto,
Difficolta, Tempo, DataCreazione, IdUtente, NumeroLike) AS(
    SELECT ricetta.*, COUNT(likes.IdRicetta) AS NumeroLike
    FROM ricetta
    LEFT JOIN likes
    ON likes.IdRicetta = ricetta.IdRicetta
    GROUP BY ricetta.IdRicetta
    LIMIT 10)
SELECT DISTINCT v1.* FROM v1
JOIN ingrediente_ricetta
ON ingrediente_ricetta.IdRicetta = v1.IdRicetta
JOIN ingrediente
ON ingrediente_ricetta.IdIngrediente = ingrediente.IdIngrediente
ORDER BY NumeroLike DESC;

WITH v1(IdRicetta, Nome, Descrizione, Passaggi, Foto,
Difficolta, Tempo, DataCreazione, IdUtente, NumeroLike) AS(
    SELECT ricetta.*, COUNT(likes.IdRicetta) AS NumeroLike
    FROM ricetta
    LEFT JOIN likes
    ON likes.IdRicetta = ricetta.IdRicetta
    JOIN ricetta_collezione
    ON ricetta_collezione.IdRicetta=ricetta.IdRicetta
    WHERE ricetta_collezione.IdCollezione = ?
    GROUP BY ricetta.IdRicetta)
SELECT DISTINCT v1.*
FROM v1
JOIN ingrediente_ricetta
ON ingrediente_ricetta.IdRicetta = v1.IdRicetta
JOIN ingrediente
ON ingrediente_ricetta.IdIngrediente = ingrediente.IdIngrediente
ORDER BY NumeroLike DESC;

SELECT * FROM ricetta
WHERE ricetta.Difficolta = ?
ORDER BY ?;

SELECT * FROM ricetta
WHERE ricetta.Tempo = ?
ORDER BY ?;

SELECT * FROM ricetta
WHERE ricetta.IdUtente = ?;

SELECT ricetta.* FROM ricetta
JOIN likes ON ricetta.IdRicetta = likes.IdRicetta
WHERE likes.IdUtente = ?;

SELECT * FROM ricetta
JOIN valutazione_ricetta
ON ricetta.IdRicetta = valutazione_ricetta.IdRicetta
WHERE valutazione_ricetta.IdUtente = ?;

SELECT ricetta.* FROM ricetta
WHERE ricetta.IdRicetta = ?;
```

```

SELECT collezione.*, categoria_nutrizionale.Nome AS NomeCategoria
FROM collezione
JOIN categoria_nutrizionale
ON categoria_nutrizionale.IdCategoria=collezione.IdCategoria
WHERE IdCollezione = ?;

SELECT * FROM collezione
WHERE collezione.IdUtente = ?;

SELECT c1.* , categoria_nutrizionale.Nome AS NomeCategoria,
ricetta.Foto AS FotoRicetta FROM collezione AS c1
JOIN categoria_nutrizionale
ON categoria_nutrizionale.IdCategoria = c1.IdCategoria
JOIN ricetta_collezione
ON ricetta_collezione.IdCollezione = c1.IdCollezione
JOIN ricetta
ON ricetta.IdRicetta = ricetta_collezione.IdRicetta
WHERE Dieta = ? AND ricetta.IdRicetta = (
    SELECT MIN(ricetta.IdRicetta)
    FROM ricetta
    JOIN ricetta_collezione
    ON ricetta_collezione.IdRicetta = ricetta.IdRicetta
    WHERE ricetta_collezione.IdCollezione = c1.IdCollezione);

SELECT utente.*,
IFNULL(temp_likes.count, 0) AS Likes,
IFNULL(temp_reviews.count, 0) AS ReviewCount,
IFNULL(temp_recipes.count, 0) AS RecipeCount,
IFNULL(temp_collections.count, 0) AS CollectionCount,
IFNULL(temp_achievements.count, 0) AS AchievementsCount
FROM utente
LEFT JOIN (
    SELECT utente.IdUtente, COUNT(*) AS count
    FROM utente JOIN likes
    ON likes.IdUtente = utente.IdUtente
    GROUP BY likes.IdUtente) AS temp_likes
ON utente.IdUtente = temp_likes.IdUtente
LEFT JOIN (
    SELECT utente.IdUtente, COUNT(*) AS count
    FROM utente JOIN valutazione_ricetta
    ON valutazione_ricetta.IdUtente = utente.IdUtente
    GROUP BY valutazione_ricetta.IdUtente) AS temp_reviews
ON utente.IdUtente = temp_reviews.IdUtente
LEFT JOIN (
    SELECT utente.IdUtente, COUNT(*) AS count
    FROM utente JOIN ricetta
    ON ricetta.IdUtente = utente.IdUtente
    GROUP BY ricetta.IdUtente) AS temp_recipes
ON utente.IdUtente = temp_recipes.IdUtente
LEFT JOIN (
    SELECT utente.IdUtente, COUNT(*) AS count
    FROM utente JOIN collezione
    ON collezione.IdUtente = utente.IdUtente
    GROUP BY collezione.IdUtente) AS temp_collections
ON utente.IdUtente = temp_collections.IdUtente
LEFT JOIN (
    SELECT utente.IdUtente, COUNT(*) AS count
    FROM utente JOIN obiettivo_ottenuto
    ON obiettivo_ottenuto.IdUtente = utente.IdUtente
    GROUP BY obiettivo_ottenuto.IdUtente) AS temp_achievements
ON utente.IdUtente = temp_achievements.IdUtente

```

```

WHERE utente.IdUtente = ?;

WITH v1(IdRicetta, Nome, Descrizione, Passaggi, Foto,
Difficolta, Tempo, DataCreazione, IdUtente, NumeroLike) AS(
    SELECT ricetta.*, COUNT(likes.IdRicetta) AS NumeroLike
    FROM ricetta LEFT JOIN likes
    ON likes.IdRicetta = ricetta.IdRicetta
    GROUP BY ricetta.IdRicetta)
SELECT DISTINCT v1.* FROM v1
JOIN ingrediente_ricetta
ON ingrediente_ricetta.IdRicetta = v1.IdRicetta
JOIN ingrediente
ON ingrediente_ricetta.IdIngrediente = ingrediente.IdIngrediente
ORDER BY NumeroLike DESC
LIMIT 3;

WITH v1(IdRicetta, Nome, Descrizione, Passaggi, Foto,
Difficolta, Tempo, DataCreazione, IdUtente, NumeroLike) AS(
    SELECT ricetta.*, COUNT(likes.IdRicetta) AS NumeroLike
    FROM ricetta LEFT JOIN likes
    ON likes.IdRicetta = ricetta.IdRicetta
    GROUP BY ricetta.IdRicetta)
SELECT DISTINCT v1.* FROM v1
JOIN ingrediente_ricetta
ON ingrediente_ricetta.IdRicetta=v1.IdRicetta
JOIN ingrediente
ON ingrediente.IdIngrediente=ingrediente_ricetta.IdIngrediente
WHERE v1.Nome LIKE "%?%"
AND ingrediente.IdIngrediente IN (
    SELECT IdIngrediente FROM ingrediente WHERE IdCategoria = ?);

WITH v1(IdRicetta, Nome, Descrizione, Passaggi, Foto,
Difficolta, Tempo, DataCreazione, IdUtente, NumeroLike) AS(
    SELECT ricetta.*, COUNT(likes.IdRicetta) AS NumeroLike
    FROM ricetta LEFT JOIN likes
    ON likes.IdRicetta = ricetta.IdRicetta
    GROUP BY ricetta.IdRicetta)
SELECT DISTINCT v1.* FROM v1
JOIN ingrediente_ricetta
ON ingrediente_ricetta.IdRicetta=v1.IdRicetta
JOIN ingrediente
ON ingrediente.IdIngrediente=ingrediente_ricetta.IdIngrediente
WHERE v1.Nome LIKE "%?%"
AND ingrediente.IdIngrediente IN (
    SELECT IdIngrediente FROM ingrediente WHERE IdCategoria = ?)
AND ingrediente.IdIngrediente NOT IN (
    SELECT ingrediente.IdIngrediente
    FROM ingrediente
    JOIN ingrediente_categoria
    ON ingrediente_categoria.IdIngrediente=ingrediente.IdIngrediente
    WHERE IdCategoria != ?);

SELECT * FROM ingrediente
WHERE LOWER(Nome) LIKE "%?%";

SELECT categoria_nutrizionale.*
FROM categoria_nutrizionale

SELECT DISTINCT categoria_nutrizionale.*
FROM categoria_nutrizionale
JOIN ingrediente_categoria
ON categoria_nutrizionale.IdCategoria = ingrediente_categoria.IdCategoria

```

```

JOIN ingrediente
ON ingrediente_categoria.IdIngrediente = ingrediente.IdIngrediente
JOIN ingrediente_ricetta
ON ingrediente_ricetta.IdIngrediente = ingrediente.IdIngrediente
WHERE ingrediente_ricetta.IdRicetta = ?;

SELECT categoria_nutrizionale.*
FROM categoria_nutrizionale
JOIN obiettivo
ON obiettivo.IdCategoria=categoria_nutrizionale.IdCategoria
JOIN obiettivo_ottenuto
ON obiettivo_ottenuto.IdObiettivo=obiettivo.IdObiettivo
WHERE categoria_nutrizionale.IdCategoria != 1
AND IdUtente = ?;

SELECT IFNULL(SUM(CASE WHEN Voto = 1 THEN 1 ELSE 0 END), 0)
AS VotiPositivi,
IFNULL(SUM(CASE WHEN Voto = 0 THEN 1 ELSE 0 END), 0)
AS VotiNegativi
FROM valutazione_ricetta
WHERE IdRicetta = ?;

SELECT IFNULL(SUM(CASE WHEN Voto = 1 THEN 1 ELSE 0 END), 0)
AS VotiPositivi,
IFNULL(SUM(CASE WHEN Voto = 0 THEN 1 ELSE 0 END), 0)
AS VotiNegativi
FROM valutazione_collezione
WHERE IdCollezione = ?;

SELECT valutazione_ricetta.IdUtente, IdRicetta, Voto, DataValutazione,
Commento, utente.Username AS NomeUtente, utente.Foto AS FotoUtente
FROM valutazione_ricetta
JOIN utente ON utente.IdUtente=valutazione_ricetta.IdUtente
WHERE IdRicetta = ?
ORDER BY DataValutazione DESC;

SELECT valutazione_collezione.IdUtente, IdCollezione, Voto, DataValutazione,
Commento, utente.Username AS NomeUtente, utente.Foto AS FotoUtente
FROM valutazione_collezione
JOIN utente ON utente.IdUtente=valutazione_collezione.IdUtente
WHERE IdCollezione = ?
ORDER BY DataValutazione DESC;

SELECT * FROM valutazione_ricetta
WHERE IdUtente = ? AND IdRicetta = ? AND Voto = ?;

SELECT * FROM valutazione_ricetta
WHERE IdUtente = ? AND IdRicetta = ? AND Voto != ?;

DELETE FROM valutazione_ricetta
WHERE IdUtente = ? AND IdRicetta = ?;

INSERT INTO valutazione_ricetta
(IdUtente, IdRicetta, Voto, DataValutazione, Commento)
VALUES (?, ?, ?, NOW(), '');

SELECT * FROM valutazione_collezione
WHERE IdUtente = ? AND IdCollezione = ? AND Voto = ?;

SELECT * FROM valutazione_collezione

```

```

WHERE IdUtente = ? AND IdCollezione = ? AND Voto != ?;

DELETE FROM valutazione_collezione
WHERE IdUtente = ? AND IdCollezione = ?;

INSERT INTO valutazione_collezione
(IdUtente, IdCollezione, Voto, DataValutazione, Commento)
VALUES (?, ?, ?, NOW(), '');

SELECT * FROM valutazione_ricetta
WHERE IdUtente = ? AND IdRicetta = ?;

SELECT * FROM obiettivo JOIN obiettivo_ottenuto
ON obiettivo.IdObiettivo = obiettivo_ottenuto.IdObiettivo
JOIN collezione
ON collezione.IdCategoria = obiettivo.IdCategoria
WHERE obiettivo_ottenuto.IdUtente = ?
AND collezione.IdCollezione = ?
AND collezione.IdCategoria = obiettivo.IdCategoria;

UPDATE valutazione_ricetta
SET Commento = ?
WHERE IdUtente = ? AND IdRicetta = ?;

SELECT * FROM valutazione_collezione
WHERE IdUtente = ? AND IdCollezione = ?;

UPDATE valutazione_collezione
SET Commento = ?
WHERE IdUtente = ? AND IdCollezione = ?;

WITH v1(IdRicetta, Nome, Descrizione, Passaggi, Foto,
Difficoltà, Tempo, DataCreazione, IdUtente, NumeroLike) AS(
    WITH v2(IdRicetta, Nome, Descrizione, Passaggi, Foto,
    Difficoltà, Tempo, DataCreazione, IdUtente, NumeroLike) AS(
        WITH v3(IdRicetta, Nome, Descrizione, Passaggi, Foto,
        Difficoltà, Tempo, DataCreazione, IdUtente, NumeroLike) AS(
            SELECT ricetta.*, COUNT(likes.IdRicetta) AS NumeroLike
            FROM ricetta
            LEFT JOIN likes ON likes.IdRicetta = ricetta.IdRicetta
            GROUP BY ricetta.IdRicetta)
        SELECT DISTINCT v3.* FROM v3
        JOIN ingrediente_ricetta
        ON ingrediente_ricetta.IdRicetta = v3.IdRicetta
        JOIN ingrediente
        ON ingrediente_ricetta.IdIngrediente = ingrediente.IdIngrediente
        JOIN ingrediente_categoria
        ON ingrediente.IdIngrediente = ingrediente_categoria.IdIngrediente
        JOIN categoria_nutrizionale
        ON categoria_nutrizionale.IdCategoria = ingrediente_categoria.IdCategoria
        WHERE LOWER(categoria_nutrizionale.Nome) LIKE "%?%""
        OR LOWER(ingrediente.Nome) LIKE "%?%""
        OR LOWER(v3.Nome) LIKE "%?%""
        LIMIT 10)
    SELECT v2.* FROM v2
    WHERE v2.Difficoltà = 1
    AND FLOOR(v2.Tempo / 10) * 10 = FLOOR(5 / 10) * 10
    GROUP BY v2.IdRicetta)
SELECT DISTINCT v1.* FROM v1

```

```

JOIN ingrediente_ricetta
ON ingrediente_ricetta.IdRicetta = v1.IdRicetta
JOIN ingrediente
ON ingrediente.IdIngrediente = ingrediente_ricetta.IdIngrediente
JOIN ingrediente_categoria
ON ingrediente.IdIngrediente = ingrediente_categoria.IdIngrediente
JOIN categoria_nutrizionale
ON categoria_nutrizionale.IdCategoria = ingrediente_categoria.IdCategoria
WHERE categoria_nutrizionale.Nome IN ('Generico')

INSERT INTO ricetta (Nome, Descrizione, Passaggi, Foto,
                     Difficolta, Tempo, DataCreazione, IdUtente)
VALUES (?, ?, ?, ?, ?, ?, ?, NOW(), ?);

INSERT INTO ingrediente_ricetta
(IdIngrediente, IdRicetta, PesoInGrammi)
VALUES (?, ?, ?);

SELECT MAX(IdRicetta) AS IdRicetta
FROM ricetta WHERE IdUtente = ?;

INSERT INTO collezione
(Nome, Descrizione, Dieta, DataCreazione, IdUtente, IdCategoria)
VALUES (?, ?, ?, ?, NOW(), ?, ?);

INSERT INTO ricetta_collezione (IdRicetta, IdCollezione)
VALUES (?, ?);

SELECT MAX(IdCollezione) AS IdCollezione
FROM collezione
WHERE IdUtente = ?;

SELECT * FROM utente WHERE Email = ? AND Pwd = ?;

SELECT * FROM utente WHERE Email = ?;

INSERT INTO utente
(Username, Pwd, Email, Foto, Esperienza, Livello)
VALUES (?, ?, ?, ?, 100, 1);

INSERT INTO obiettivo_ottenuto
(IdObiettivo, IdUtente, DataOtttenimento)
VALUES (1, ?, CURDATE());

SELECT * FROM likes
WHERE IdRicetta = ? AND IdUtente = ?;

DELETE FROM likes
WHERE IdRicetta = ? AND IdUtente = ?;

INSERT INTO likes
(IdRicetta, IdUtente, Data)
VALUES (?, ?, CURDATE());

SELECT * FROM ricetta
JOIN likes ON likes.IdRicetta = ricetta.IdRicetta
WHERE likes.IdUtente = ? AND ricetta.IdRicetta = ?;

WITH v2 AS (
    WITH v1 AS (
        SELECT collezione.*,

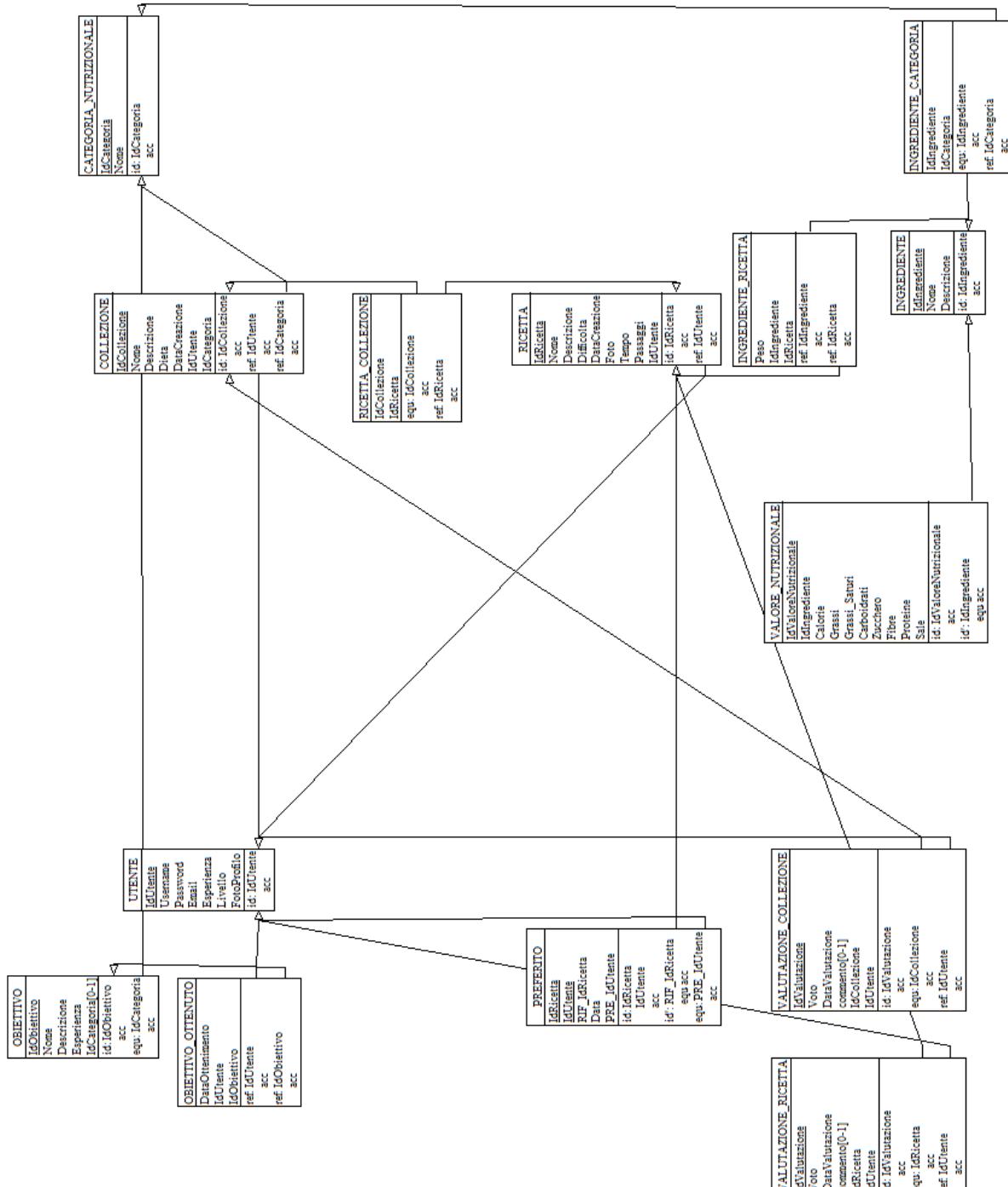
```

```

        categoria_nutrizionale.Nome AS NomeCategoria
    FROM collezione JOIN categoria_nutrizionale
    ON categoria_nutrizionale.IdCategoria = collezione.IdCategoria
    WHERE collezione.Dieta = ?
    AND collezione.DataCreazione = ?
    AND categoria_nutrizionale.Nome = ?)
SELECT v1.*,
       avg(ricetta.Difficoltà) AS avDiff,
       COUNT(ricetta.Difficoltà) AS num
FROM v1
JOIN ricetta_collezione
ON ricetta_collezione.IdCollezione = v1.IdCollezione
JOIN ricetta
ON ricetta.IdRicetta = ricetta_collezione.IdRicetta
WHERE LOWER(ricetta.Nome) LIKE "%?%" OR v1.Nome LIKE "%?%"
GROUP BY v1.IdCollezione
ORDER BY ?)
SELECT v2.*, ricetta.Foto AS FotoRicetta
FROM v2
JOIN ricetta_collezione
ON ricetta_collezione.IdCollezione = v2.IdCollezione
JOIN ricetta
ON ricetta.IdRicetta = ricetta_collezione.IdRicetta
WHERE ricetta_collezione.IdRicetta = (
    SELECT MIN(ricetta.IdRicetta)
    FROM ricetta JOIN ricetta_collezione
    ON ricetta_collezione.IdRicetta = ricetta.IdRicetta
    WHERE ricetta_collezione.IdCollezione = v2.IdCollezione)
AND v2.num = ? AND v2.avDiff = ?;

WITH v1 AS(
    SELECT DISTINCT obiettivo.*,
                   obiettivo_ottenuto.DataOttenimento,
                   obiettivo_ottenuto.IdUtente
    FROM obiettivo
    LEFT JOIN obiettivo_ottenuto
    ON obiettivo_ottenuto.IdObiettivo = obiettivo.IdObiettivo)
SELECT v1.* FROM v1
WHERE IdUtente IS NULL OR IdUtente = ?;

```



Capitolo 4

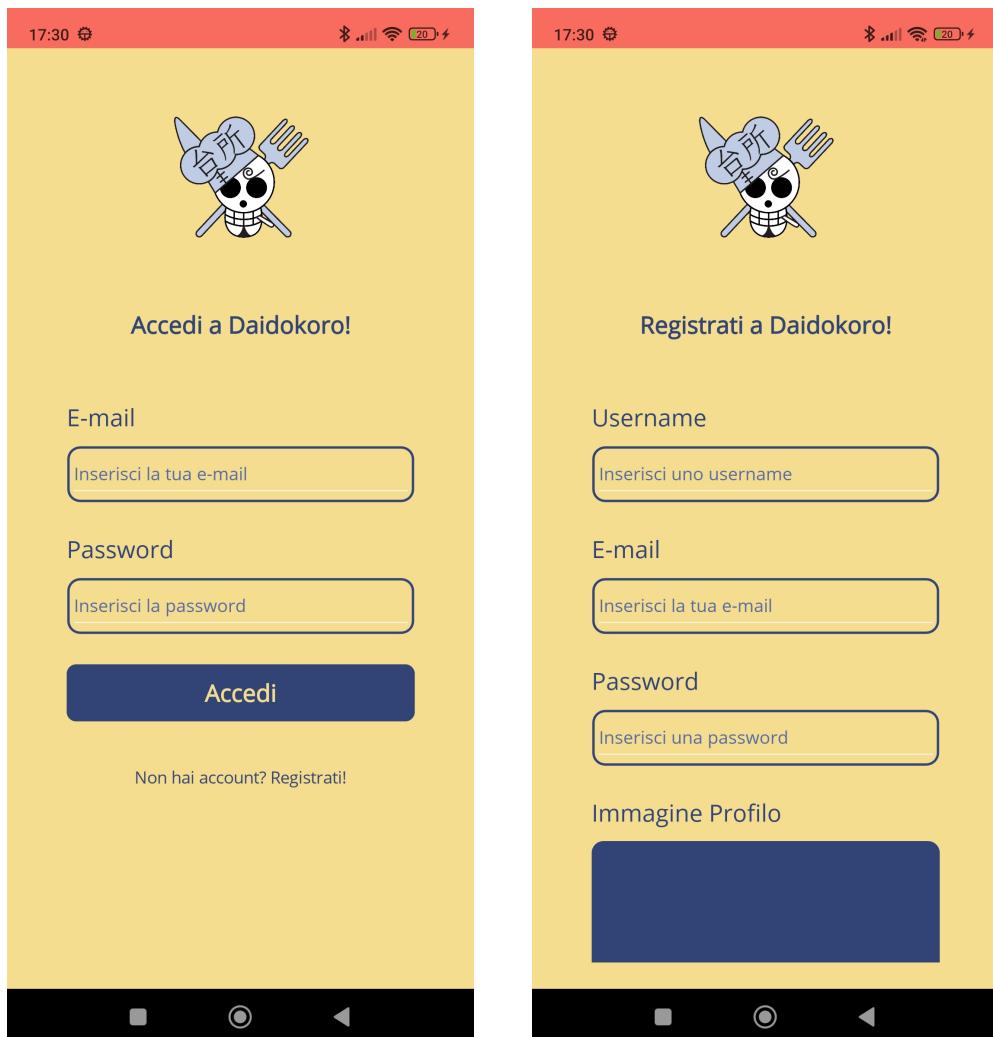
Progettazione applicazione

4.1 Descrizione dell'architettura dell'applicazione realizzata

Abbiamo sviluppato un'applicazione Social Media di ricette culinarie per dispositivi mobile in linguaggio C# utilizzando il framework .NET MAUI (utilizzato per la creazione di applicazioni mobile). Per le connessioni al database viene utilizzata la libreria C# MySQLConnector e il database risiede in un server MySQL. Per ogni tabella contenuta nel DB viene creata una corrispettiva classe che la rappresenta all'interno del model dell'applicazione. Inoltre abbiamo creato una classe DBService che gestisce le connessioni effettuate al database e presenta vari metodi:

- TryConnection(DBCredentials dbc): prende in input le credenziali per accedere al database e restituisce un booleano sul risultato della connessione al database.
- ExistInTable(string query): restituisce un booleano in base al numero di elementi che produce la query data (0 elementi prodotti = false)
- GetData(string query): restituisce una lista di elementi generici in base alla tabella su cui viene effettuata la query e al tipo specificato.
- InsertElement(List<Tuple<string, object>> values, string query): inserisce gli elementi nella tabella specificata nella query inserendo nei valori della query non settati (?) il rispettivo valore contenuto nella lista di tuple.
- RemoveOrUpdateElement(string query): rimuove o aggiorna la riga di una tabella seguendo la query data.

All'avvio dell'applicazione, se si tratta del primo avvio, comparirà la schermata di Login e Registrazione.



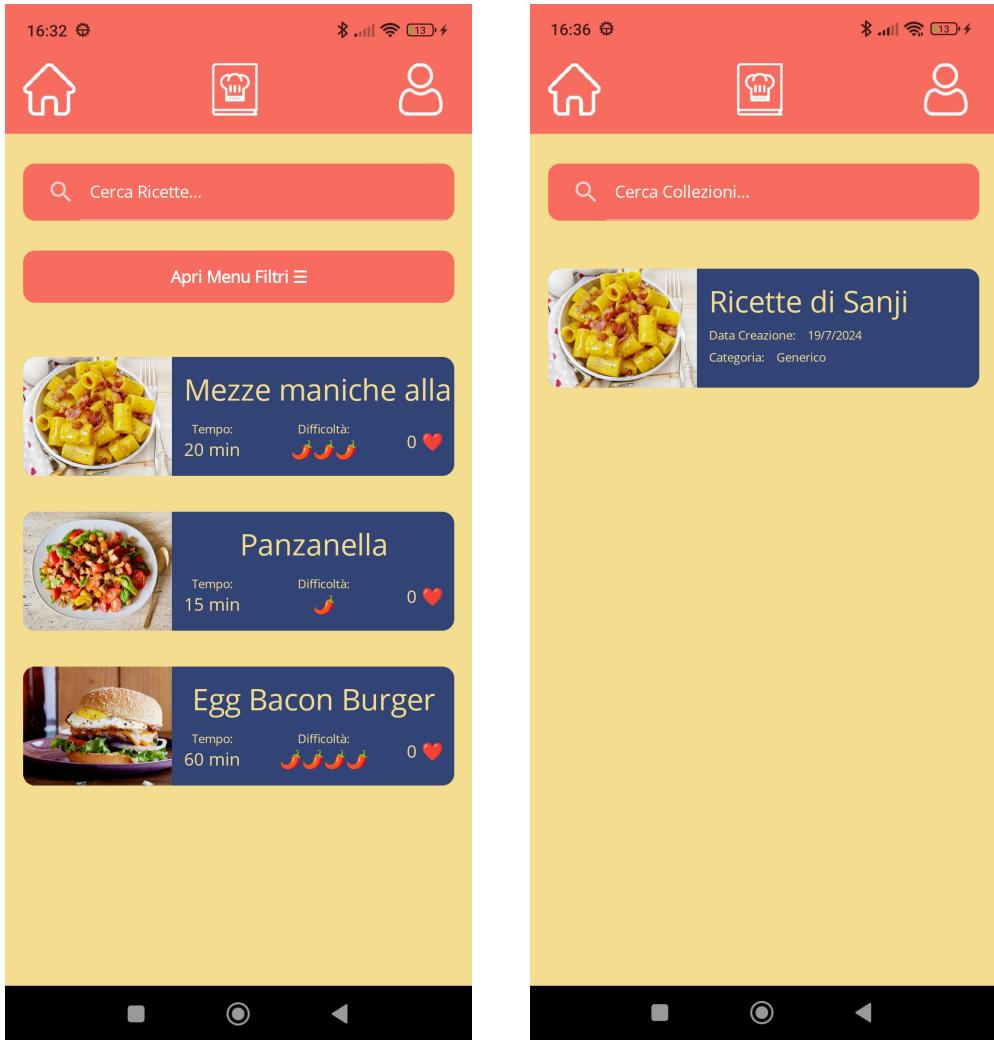
Dopo essersi registrati e aver effettuato l'accesso verrà memorizzato nei dati dell'applicazione installata l'identificativo dell'utente che ha effettuato l'accesso in modo da poter reindirizzare l'utente sulla schermata principale ad ogni avvio successivo dell'app e caricare i suoi dati.

Nella Home Page è possibile visualizzare le 3 ricette con più like.
Clickando, invece, sull'icona centrale del menu di navigazione sovrastante, si potrà andare sulla pagina di navigazione delle ricette, collezioni e diete



Successivamente nella pagina di navigazione in base all'elemento clickato comparirà la pagina di visualizzazione della lista del rispetto elemento.

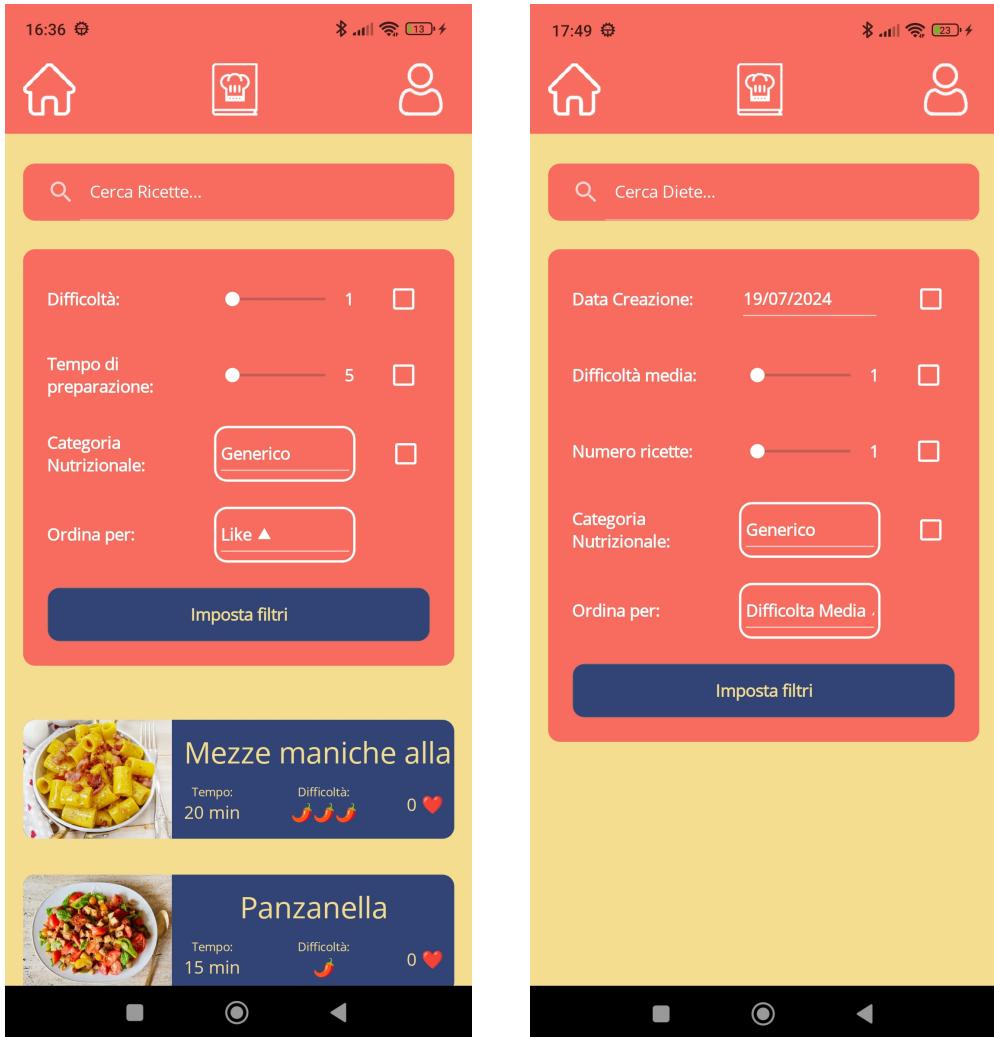
Di seguito si ha a sinistra la pagina con la lista delle ricette di qualunque utente con la rispettiva barra di ricerca e filtri. Mentre a destra la pagina con la lista di collezioni di qualunque utente e la barra di ricerca.



Ogni ricetta nella lista presenta l'immagine, il nome, il tempo di preparazione, difficoltà rappresentata da 1 a 5 con l'emoji di un peperoncino e il numero di like ricevuti da utenti. Mentre le collezioni avranno come immagine quella della ricetta con più like nella collezione, la data di creazione e la categoria.

La pagina delle diete è composta come quella delle collezioni ma presente un menu dei filtri come quello delle ricette oltre alla barra di ricerca.

La barra di ricerca delle diete effettuerà la ricerca di una ricetta in base al nome, al nome degli ingredienti associati e al nome della categoria nutrizionale degli ingredienti, mentre quella delle collezioni e delle diete cercherà in base al nome della collezione/dieta e al nome delle ricette contenute.



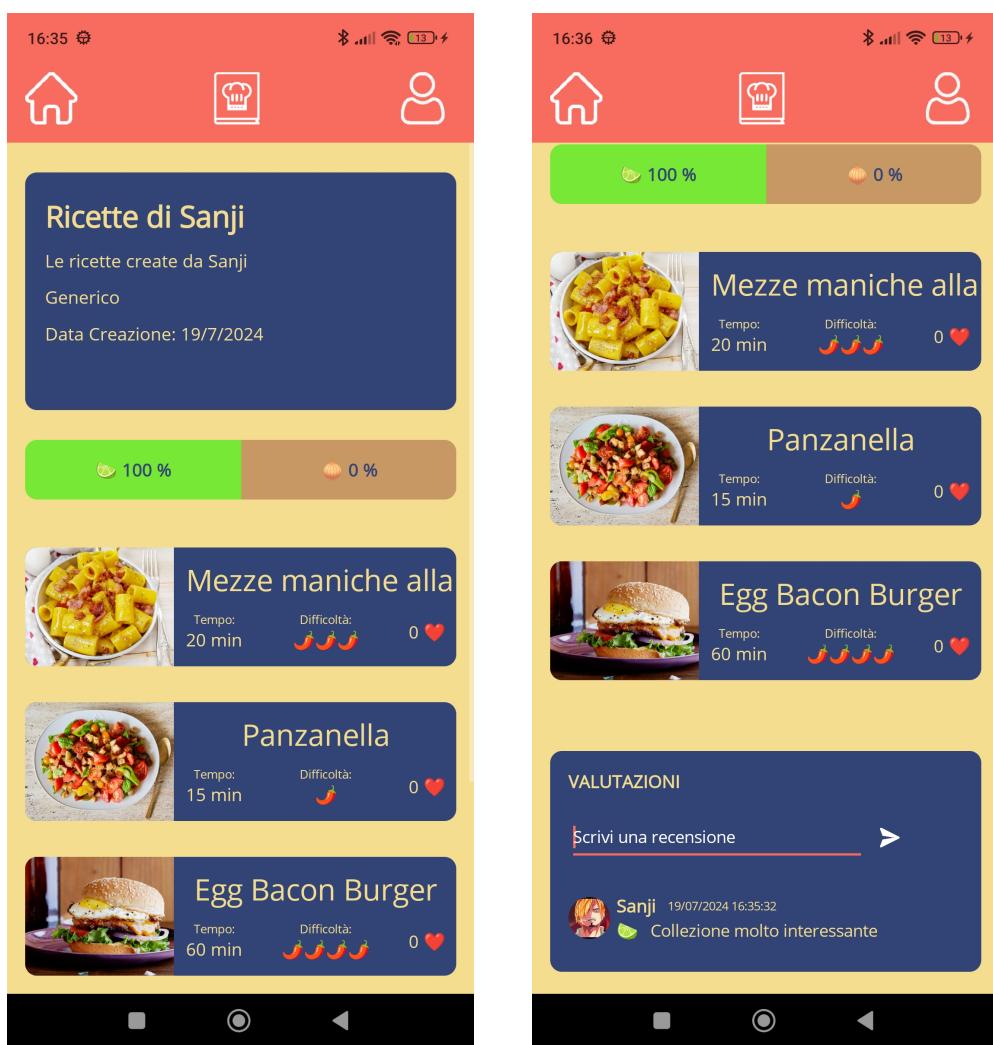
Per quanto riguarda le ricette, il menu dei filtri contiene uno slider per la difficoltà e uno per il tempo di preparazione, un selettore per la categoria nutrizionale e un selettore per l'ordinamento da effettuare sulle ricette filtrate. Mentre il menu dei filtri delle diete contiene un selettore per la data di creazione, uno slider per la difficoltà media delle ricette contenute nella dieta, uno slider per il numero massimo di ricette nella dieta e i due selettori per la categoria nutrizionale della dieta e l'ordinamento sulle diete filtrate.

Quando si va su una delle pagine contenenti le liste di elementi, clickando su un elemento si andrà alla pagina del singolo elemento con tutte le informazioni specifiche. Di seguito si ha la pagina di una ricetta.



In questa pagina l'utente potrà visualizzare tutte le informazioni della ricetta come la descrizione, gli ingredienti e i passaggi. Inoltre potrà aggiungere la ricetta ai preferiti clickando sul pulsante con il cuore e potrà valutare la ricetta con un voto positivo (**Lime**) o negativo (**Cipolla**). In fondo alla pagina l'utente troverà la sezione delle valutazioni dove compariranno tutte le valutazioni degli utenti e l'utente potrà aggiungere un commento alla valutazione effettuata in precedenza.

Nella pagina della singola collezione/dieta, invece, si ha inizialmente una sezione con le informazioni principali, seguita dagli stessi bottoni per le valutazioni come nella pagina delle ricette e la lista delle ricette contenute nella collezione/dieta. In fondo alla pagina si ha la sezione delle valutazioni con la possibilità di inserire il commento come per le ricette, nella pagina di una dieta però il commento può essere inserito solo se si ha sbloccato l'obiettivo della categoria nutrizionale a cui appartiene la dieta.

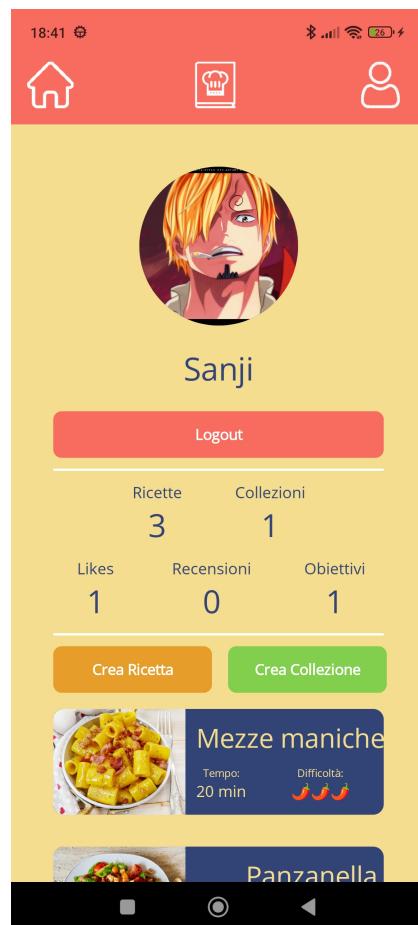


Tornando sul menu sovrastante di navigazione, clickando sull'icona a destra si andrà sulla propria pagina utente.

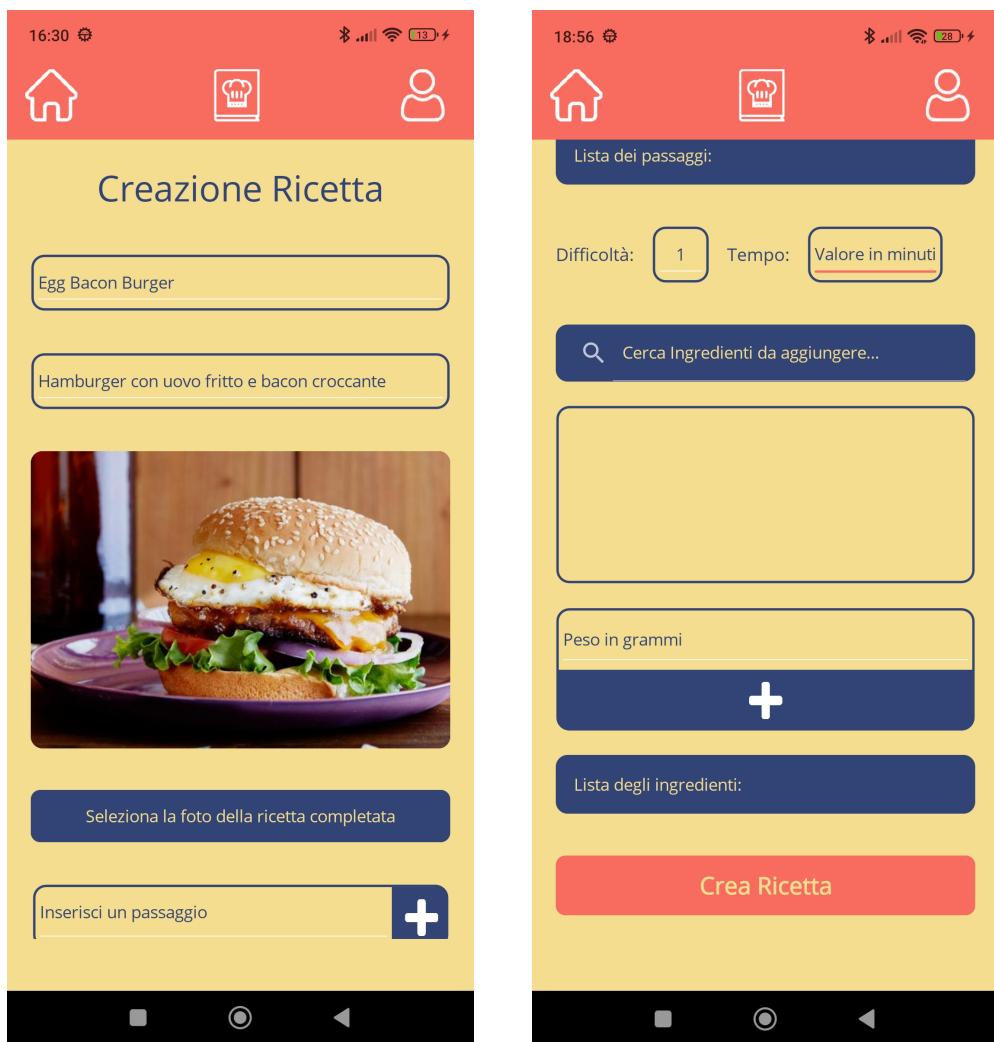
In questa pagina si ha la foto profilo inserita in fase di registrazione e il nome, seguiti da il tasto per effettuare il logout e i seguenti contatori delle statistiche dell'utente:

- Numero di ricette create
- Numero di collezioni create
- Numero di recensioni create
- Numero di ricette aggiunte ai preferiti
- Numero di obiettivi ottenuti

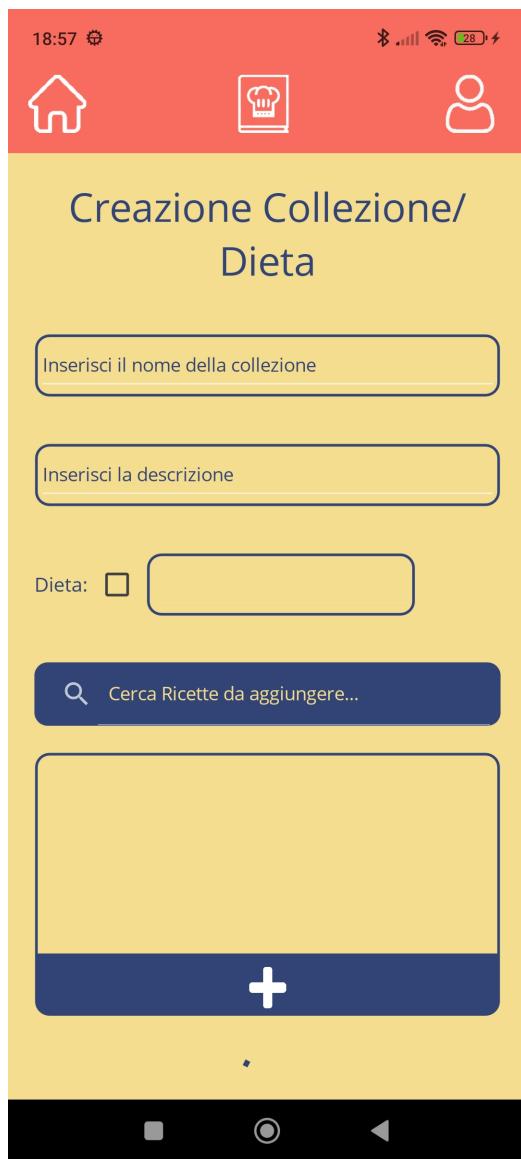
Infine si hanno i pulsanti per la creazione di ricette e collezioni e una lista dove, al click di un contatore, compariranno gli elementi a cui fa riferimento quel contatore.



Clickando sul pulsante per la creazione di una ricetta comparirà la suddetta pagina con i vari campi da completare: **nome**, **descrizione**, **foto** (selezionandola dal proprio dispositivo), **lista dei passaggi** (scrivendo un passaggio alla volta e aggiungendolo alla lista), **difficoltà**, **tempo** e la **lista degli ingredienti** con la relativa quantità in grammi.



Mentre clickando su quello per la creazione di collezioni/diete comparirà la pagina con i seguenti campi da completare: **nome**, **descrizione**, checkbox per segnalare se si sta creando una **dieta** e scegliere la categoria nutrizionale e la **lista delle ricette** da aggiungere.



Tutti i metodi che eseguono delle query sono contenuti dentro la classe Main-ViewModel che a sua volta utilizza un'istanza globale di DBService per eseguire le query.

4.2 Elenco delle operazioni

4.2.1 Operazioni di selezione

Nome	Operazione
GetIngredients(int IdRicetta)	Restituisce tutti gli ingredienti appartenenti alla ricetta dell'ID dato
GetRecipes()	Restituisce le prime 10 ricette dotate del numero di like
GetRecipesByCollection(int IdCollezione)	Restituisce tutte le ricette appartenenti alla collezione dell'ID dato
GetRecipesByDifficulty(int difficulty, string orderby)	Restituisce tutte le ricette con una data difficoltà e ordinate con il dato ordinamento
GetRecipesByTime(int time, string orderby)	Restituisce tutte le ricette con una data tempo di preparazione e ordinate con il dato ordinamento
GetRecipesByUser(int userID)	Restituisce tutte le ricette appartenenti ad un dato utente
GetLikedRecipes(int userID)	Restituisce tutte le ricette aggiunte ai preferiti da un dato utente
GetReviewedRecipes(int userID)	Restituisce tutte le ricette valutate da un dato utente
GetRecipeById(int IdRicetta)	Restituisce la ricetta con il dato ID
GetCollectionById(int IdCollezione)	Restituisce la collezione con il dato ID compresa del nome della categoria nutrizionale a cui appartiene
GetCollectionsByUser(int userID)	Restituisce tutte le collezioni appartenenti ad un dato utente
GetCollectionsOrDiets(int Dieta)	Restituisce tutte le collezioni o diete, in base al parametro passato, comprese del nome della categoria nutrizionale a cui appartengono e della foto della ricetta con identificato meno recente che vi appartiene
GetUserById(int id)	Restituisce l'utente con ID dato compreso dei contatori di: ricette preferite, ricette valutate, ricette create, collezioni create e obiettivi ottenuti
GetMonthRecipes()	Restituisce le 3 ricette con più like

Nome	Operazione
GetNutritionalCategories(int IdRicetta)	Restituisce tutte le categorie nutrizionali a cui appartengono gli ingredienti di una ricetta
GetUnlockedNutritionalCategories()	Restituisce le categorie nutrizionali sbloccate dall'utente attraverso gli obiettivi
GetRatingsCountGroupByVoto(int id, bool isRecipe)	Restituisce il numero di voti positivi e negativi di una ricetta o di una collezione
GetRatingsById(int id, bool isRecipe)	Restituisce le valutazioni di una ricetta o di una collezione comprese dei nomi degli utenti e delle foto profilo
GetSearchedRecipes(string name, int IdCategoria)	Restituisce tutte le ricette con nome simile a quello dato e con tutti gli ingredienti appartenenti alla categoria nutrizionale data
GetSearchedIngredients(string text)	Restituisce tutti gli ingredienti con nome simile a quello dato
GetNutritionalCategories()	Restituisce tutte le categorie nutrizionali
GetFilteredRecipes(string difficulty, string time, string orderby, string category, string text)	Restituisce le ricette filtrate per difficoltà, tempo di preparazione, categoria nutrizionale degli ingredienti, nome e le ordina in base all'ordinamento dato
GetInsertedRecipeId()	Restituisce l'ID dell'ultima ricetta inserita dall'utente corrente
GetInsertedCollectionId()	Restituisce l'ID dell'ultima collezione inserita dall'utente corrente
CanUserLogin(string email, string password)	Restituisce un booleano in base al numero di occorrenze nella tabella utente con email e password pari a quelle date (0 = false)
GetLoggedUserId(string email)	Restituisce l'ID dell'utente che ha l'email pari a quella data
CanUserRegister(string email)	Restituisce un booleano in base al numero di occorrenze nella tabella utente con email pari a quella data (0 = true)
IsRecipeLikedByUser(int IdRicetta)	Restituisce un booleano in base al numero di occorrenze nella tabella likes ID della ricetta pari a quella dato (0 = false)
GetFilteredCollections(string text, string difficulty, string date, string orderby, string recipeNumber, int dieta, string nutritionalCategory)	Restituisce le collezioni o diete filtrate per difficoltà, data di creazione, categoria nutrizionale, nome, numero di ricette e le ordina in base all'ordinamento dato
GetObjectives(int IdUtente)	Restituisce gli obiettivi dell'utente corrente

4.2.2 Operazioni di inserimento, aggiornamento e rimozione

InsertRating(List<Tuple<string, object>> rating, bool isRecipe)	Inserisce una valutazione su una ricetta o su una collezione da parte dell'utente corrente. Se l'elemento è già stato valutato e la valutazione da voler inserire è l'opposto di quella corrente, verrà prima eliminata la vecchia valutazione e poi verrà inserita quella attuale
InsertNewRecipe(List<Tuple<string, object>> recipe)	Inserisce una nuova ricetta da parte dell'utente corrente
InsertRecipeIngredient(List<Tuple<string, object>> ingredientRecipe)	Collega gli ingredienti ad una ricetta appena creata con i relativi pesi
InsertNewCollection(List<Tuple<string, object>> collection)	Inserisce una nuova collezione da parte dell'utente corrente
InsertCollectionRecipe(List<Tuple<string, object>> recipeCollection)	Collega le ricette ad una collezione appena creata
RegisterUser(List<Tuple<string, object>> userData)	Inserisce un nuovo utente
InsertReviewIfRatedByUser(int id, string comment, bool isRecipe)	Aggiorna, se esistente, una valutazione su una ricetta o su una collezione, inserendo il commento scritto dall'utente corrente
GiveRegisterObjective()	Sblocca all'utente che si è appena registrato l'obiettivo iniziale per la registrazione
AddOrRemoveRecipeFromLiked(int IdRicetta)	Aggiunge o rimuove dai preferiti dell'utente corrente una ricetta in base allo stato attuale