CustomCooking

UN'APPLICAZIONE *MOBILE SOCIAL* DI RICETTE CHE TI AIUTA A SCEGLIERE COSA PREPARARE IN BASE A QUELLO CHE HAI IN CUCINA

Gozzoli Alessio
0000666684
alessio.gozzoli@studio.unibo.it

Abrignani Federico
0000671199
federico.abrignani@studio.unibo.it

Ognibene Tommaso
0000672192
tommaso.ognibene@studio.unibo.it

Table of Contents

1 Analisi dei requisiti

1.1 Requisiti espressi in linguaggio naturale

Si vuole realizzare una base di dati per un'applicazione *mobile* di ricette.

L'applicazione *mobile* deve permettere di scegliere una ricetta sulla base delle disponibilità dell'utente, in termini di prodotti alimentari e utensili da cucina.

La base di dati si compone delle entità *Recipe*, *Product*, *Tool*, *Account*, *Step*, e delle relazioni *Ingredient*, *Author*, *Rating*, *ToolAvailability*, *ProductAvailability*, *ToolSet*, *Sequence*.

Per quanto concerne *Recipe*, ci interessa rappresentare un identificatore univoco (*idRecipe*), nome, tipo, descrizione, cucina a cui appartiene, eventuale origine regionale, tempo stimato di preparazione, difficoltà secondo l'autore, difficoltà secondo le valutazioni degli utenti utilizzatori, eventuale nome e riferimento ad autore esterno della ricetta (ad esempio l'autore del libro di ricette dalla quale la ricetta è stata estrapolata).

Per quanto concerne *Product*, ci interessa rappresentare nome, classe del prodotto, eventuale URL di un'immagine didascalica.

Per quanto concerne *Tool*, ci interessa rappresentare nome ed eventuale URL di un'immagine didascalica.

Per quanto concerne *Account*, ci interessa rappresentare username, email, password, avatar e data di registrazione.

Per quanto concerne *Step*, ci interessa rappresentare un identificatore numerico, descrizione, tempo stimato, eventuale URL di un'immagine didascalica, eventuale URL di un video esplicativo.

Gli ingredienti per una ricetta possono essere opzionali o necessari, lo stesso vale per gli utensili.

1.2 Glossario dei termini

TERMINE	DESCRIZIONE	SINONIMI	COLLEGAMENTI
Account	Utente registrato	**User**	Recipe, Product, Tool, Author
Recipe	Algoritmo grastronomico		Account, Step, Tool, Product
Product	Prodotto alimentare		Account, Recipe
Tool	Strumento per cucinare		Account, Recipe
Step	Passo atomico di una ricetta		Recipe
Ingredient	Prodotto alimentare relativo ad una ricetta		Product, Recipe
Sequence	Sequenza di step per preparare una ricetta		Recipe
ProductAvalaibilty	Disponibilità di un prodotto da parte di un utente		Account, Product
ToolAvailability	Disponibilità di uno strumento da parte di un utente		Account, Tool
Rating	Valutazione di una ricetta da parte di un utente		Account, Recipe
Author	Autore di una ricetta		Account, Recipe
ToolSet	Strumento relativo ad una ricetta		Tool, Recipe

1.3 Strutturazione dei requisiti

- Frasi di carattere generale:
  Si vuole costruire una base di dati per un'applicazione mobile che consente di scegliere una ricetta sulla base delle proprie disponibilità in termini di prodotti e utensili, filtrando in termini di tempo, difficoltà e valutazione.
- Frasi relative a *Recipe*:
  Per le ricette, si intende rappresentare un identificatore numerico, nome, tipo, descrizione, cucina di appartenenza, origine geografica, tempo di preparazione, difficoltà secondo l'autore, difficoltà media secondo gli utenti, valutazione media secondo gli utenti, eventuale nome di un autore esterno, eventuale link alla fonte esterna.
- Frasi relative a *Product*:
  Per i prodotti alimentari, si intende rappresentare nome, classe, ed eventuale immagine didascalica.
- Frasi relative a *Tool*:
  Per gli utensili, si intende rappresentare nome ed eventuale immagine didascalica.
- Frasi relative a *Account*:
  Per gli utenti, si intende rappresentare username, email, password, avatar e data di registrazione.
- Frasi relative a *Step*:
  Per il passo esecutivo di una ricetta, si intende rappresentare un identificatore numerico, descrizione, tempo stimato, eventuale immagine didascalica ed eventuale video esplicativo.
- Frasi relative ai componenti di una ricetta:
  Occorre rappresentare:
  - Gli ingredienti (necessari o opzionali),
  - Gli utensili (necessari o opzionali),
  - La sequenza dei passi esecutivi.
- Frasi relative alla provenienza e valutazione di una ricetta:
  Occorre rappresentare:
  - L'autore della ricetta,
  - La valutazione (voto, commento, difficoltà riscontrata) espressa dagli utenti.
- Frasi relative alle disponibilità di un utente:
  Occorre rappresentare:
  - Gli ingredienti disponibili,
  - Gli utensili disponibili.

1.4 Specifica operazioni

- Inserire un nuovo *Account*.
- Inserire una nuova *Recipe*.
- Inserire un nuovo *Step*.
- Inserire un nuovo *Product*.
- Inserire un nuovo *Tool*.
- Aggiungere uno *Step* ad una *Recipe*.
- Aggiungere un *Product* ad una *Recipe*.
- Aggiungere un *Tool* ad una *Recipe*.
- Aggiungere un *Product* ad un *Account*.
- Aggiungere un *Tool* ad un *Account*.
- Valutare una *Recipe*.
- Visualizzare le ricette filtrate e ordinate sulla base di parametri specifici.

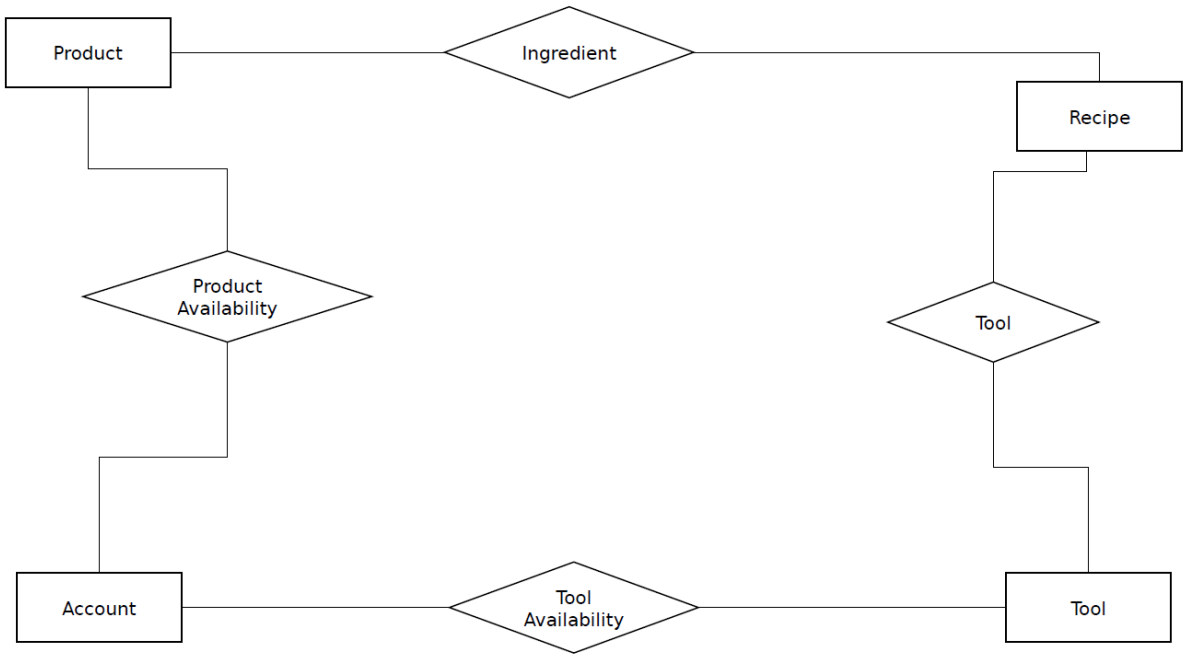
2 Progettazione concettuale

2.1 Identificazione delle entità e relazioni

Applicando la strategia *bottom*-up sono state identificate le seguenti entità e relazioni principali: *Account*, *Recipe*, *Product*, *Ingredient*, *Tool*, *Author*, *Rating*, *ProductAvailability*, *ToolAvailability* , *ToolSet*.

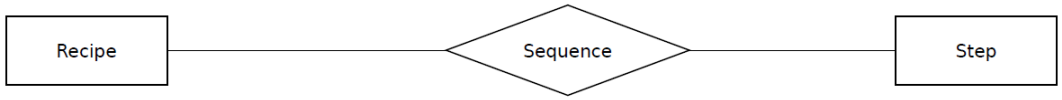
2.2 Un primo scheletro dello schema

A partire da questo elenco è stato modellato il seguento primo scheletro di schema concettuale:



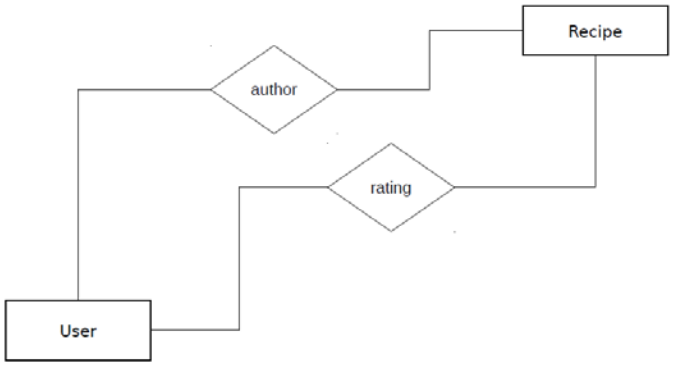
2.3 Sviluppo delle componenti dello scheletro

Ogni ricetta è costituita da uno o più passi. Di conseguenza è stato stabilito di rappresentare la ricetta mediante due differenti entità:

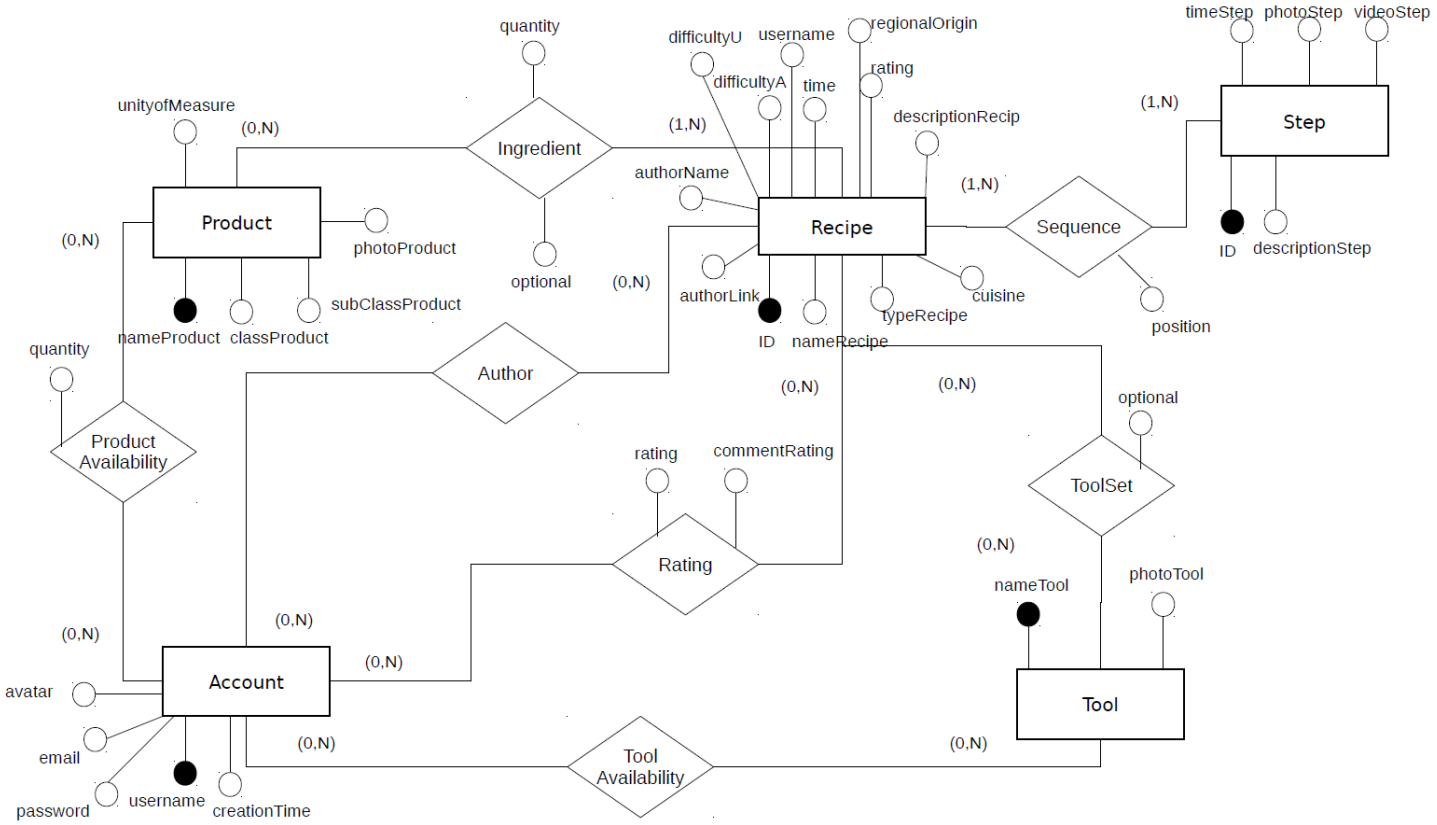


La nuova relazione (*Sequence*) è finalizzata a suddividere le ricette in fasi esecutive atomiche, affinché queste possano essere condivise, evitando ripetizioni, tra molteplici ricette. La nuova entità (*Step*) rappresenta la singola fase esecutiva.

Ogni ricetta è collegata all'utente che l'ha inserita. Si prevede inoltre la possibilità di inserire ricette create da autori noti. Inoltre, ogni utente che sceglie una ricetta può esprimere un voto, scrivere un eventuale commento ed un eventuale giudizio sulla difficoltà riscontrata.



2.4 Unione delle componenti nello schema finale ridotto



2.5 Dizionario dei dati

Entità:

NOME ENTITÀ	DESCRIZIONE	ATTRIBUTI	IDENTIFICATORE
Recipe	Ricetta	*idRecipe*, *nameRecipe*, *typeRecipe*, *descriptionRecipe*, *cuisine*, *regionalOrigin*, *preparationTime*, *difficultyAuthor*, *difficultyUsers*, *authorName*, *authorLink*, *username*, *rating*	*idRecipe*
Product	Prodotto alimentare	*nameProduct*, *classProduct*, *unityOfMeasure*, *subclassProduct*, *photoProduct*	*nameProduct*
Account	Utente registrato	*username*, *pass*, *email*, *creationTime*	*username*

Tool	Strumento per cucinare	*nameTool*, *photoTool*	*nameTool*
Step	Passo esecutivo di una ricetta	*idStep*, *descriptionStep*, *timeStep*, *photoStep*, *videoStep*	*idStep*

Relazioni:

NOME RELAZIONE	DESCRIZIONE	ENTITÀ COINVOLTE	ATTRIBUTI
Rating	Valutazione di una ricetta	Account(0,N) - Recipe(0,N)	*username, idRecipe, commentRating, rating, difficulty*
Ingredient	Ingrediente di una ricetta	Product(1,N) - Recipe(1,N)	*idRecipe, nameProduct, quantity, optional*
Sequence	Passo esecutivo di una ricetta	Recipe(1,N) - Step(1,N)	*idRecipe, idStep, position*
ToolSet	Strumento usato in una ricetta	Tool(0,N) - Recipe(0,N)	*idRecipe, nameTool, optional*
ToolAvailability	Strumento nelle disponibilità di un utente	Account(0,N) - Tool(0,N)	*username, nameTool*
ProductAvailability	Prodotto nelle disponibilità di un utente	Account(0,N) - Product(0,N)	*username, nameProduct, quantity*
Author	Autore di una ricetta	Account(0,N) - Recipe(0,N)	*idRecipe, username*

2.6 Regole aziendali

Regole di vincolo

(RV1) L'attributo *classProduct* può assumere i seguenti valori: *Meat*, *Vegetable*, *Mushrooms*, *Fruit*, *Fish*, *Spices*, *Cereals*, *Dairy product*, *Alcohol*, *Flour*.

(RV2) L'attributo *typeRecipe* può assumere i seguenti valori: *Starter*, *First course*, *Second course*, *Dessert*, *All in one*.

(RV3) L'attributo *cuisine* può assumere come valore un insieme predefinito ma estendibile di tipi di cucine.

(RV4) L'attributo *regionalOrigin* può assumere come valore un insieme predefinito ma estendibile di regioni geografiche.

(RV5) Il *rating* di una ricetta deve essere compreso tra 1 e 5.

(RV6) La *difficulty* di una ricetta deve essere compresa tra 1 e 4.

Regole di derivazione

(RD1) L'attributo *ratingUsers* di una ricetta deve essere calcolato come media dei giudizi espressi dagli utenti per quella ricetta.

(RD2) L'attributo *difficultyUsers* di una ricetta deve essere calcolato come media dei giudizi sulla difficoltà riscontrata, espressi dagli utenti per quella ricetta.

(RD3) L'attributo *preparationTime* di una ricetta deve essere calcolato come somma dei tempi stimati per ogni passo della ricetta.

3 Progettazione logica

3.1 Considerazioni sulla fase del progetto

L'applicazione *mobile* che impiegherà la base di dati sarà sviluppata come progetto del corso di Laboratorio Applicazioni Mobile, nel prossimo semestre. Di conseguenza, ad ora, non è possibile effettuare ragionamenti statistici sull'uso della base di dati.

Durante la fase di implementazione dell'applicazione, inoltre, ci si riserva la possibilità di compiere talune modifiche mirate allo schema concettuale, ove risultasse utile.

3.2 Considerazioni sui volumi stimati delle operazioni

Al fine di incrementare la scalabilità e la portabilità della base di dati, è stato fatto impiego delle *stored procedures*, in particolare in relazione alle operazioni che si stima saranno maggiormente eseguite a regime.

I benefici delle *stored procedures* in MySQL 5 sono i seguenti:

- Incapsulare funzionalità in procedure eseguibili da differenti applicazioni scritte in differenti linguaggi.
- Isolare gli utenti dalle tabelle, autorizzando l'accesso alle *stored procedures* ma non direttamente alle tabelle.
- Migliorare la *performance*.

Gli attributi *difficultyUsers*, *preparationTime*, *rating* dell'entità *Recipe* vengono generati automaticamente mediante dei *triggers*:

- *difficultyUsers* rappresenta la media delle difficoltà attribuite dagli utenti alla ricetta;
- *preparationTime* rappresenta la sommatoria dei tempi dei passi della ricetta;
- *rating* rappresenta la media dei voti attribuiti dagli utenti alla ricetta.

Tali ridondanze sono state introdotte al fine di migliorare l'efficienza delle query. Questa scelta si fonda sulla considerazione che la query che restituisce le ricette in base ai parametri di difficoltà, tempo di preparazione e valutazione, sarà una delle query maggiormente utilizzate. Di conseguenza è ragionevole calcolare una sola volta questi valori, inserendoli in attributi appositi, piuttosto che calcolarli *on-the-fly* ad ogni *call*.

3.3 Normalizzazione

Escludendo le dipendenze funzionali banali, o quelle contenenti chiavi le quali rispetterebbero comunque i vincoli di Boyce-Codd, si osserva che:

NOME ENTITÀ	COMMENTO
Account	Non contiene dipendenze funzionali.
Product	Non contiene dipendenze funzionali.
Tool	Non contiene dipendenze funzionali.
Step	Non contiene dipendenze funzionali. Occorre sottolineare che la chiave ideale sarebbe stata *descriptionStep*, tuttavia la sua dimensione supera 767 bytes, che è la dimensione massima per una chiave in InnoDB. Una soluzione percorribile poteva essere quella di impiegare come chiave l'*hash* della descrizione, ma, per evitare casi di collisioni tra *hash*, si è optato per una chiave numerica incrementale. Di conseguenza, quando l'utente tenterà di inserire un nuovo step, l'applicazione effettuerà una query per garantire l'unicità di *descriptionStep*:

```
1.  create function isStepUnique(description varchar(300)) returns boolean
2.  begin
3.      return (select count(*) from Step where descriptionStep = description) = 0;
4.  end $
5.
6.  drop procedure if exists addStep $
7.  create procedure addStep(in descriptionStep varchar(300), timeStep int, photoStep varchar
    (200), videoStep varchar(200))
8.  language sql
9.  sql security invoker
10. comment 'Add a step'
11. begin
12.     if isStepUnique(descriptionStep) then
13.         insert
14.             into Step (descriptionStep, timeStep, photoStep, videoStep)
```

	15. values (descriptionStep, timeStep, photoStep, videoStep); 16. end if; 17. end $
Recipe	Potrebbe sembrare che *authorName* e/o *authorLink* possano dipendere funzionalmente da *nameRecipe*, ma ci preme ricordare che gli utenti potranno inserire più versioni della stessa ricetta utilizzando lo stesso nome, pertanto nessun attributo può dipendere funzionalmente da *nameRecipe*.

4 Codifica SQL

4.1 Definizione dello schema

```
1.  drop database if exists CustomCooking;
2.  create database CustomCooking;
3.  use CustomCooking;
4.
5.  create table Account (
6.      username varchar(30) not null,
7.      pass varchar(32) not null,
8.      email varchar(100),
9.      creationTime datetime not null,
10.     avatar varchar(200),
11.
12.     primary key(username)
13. ) engine=InnoDB;
14.
15. create table Recipe (
16.     idRecipe int not null auto_increment,
17.     nameRecipe varchar(30) not null,
18.     typeRecipe varchar(30) not null,
19.     descriptionRecipe text,
20.     cuisine varchar(30),
21.     regionalOrigin varchar(30),
22.     preparationTime int not null,
23.     difficultyAuthor int not null,
24.     difficultyUsers double,
25.     authorName varchar(30),
26.     authorLink varchar(300),
27.     username varchar(30) not null,
28.     rating double,
29.
30.     primary key(idRecipe),
31.     foreign key(username) references Account(username)
32. ) engine=InnoDB;
33.
34. create table Step (
35.     idStep int not null auto_increment,
36.     descriptionStep varchar(300) not null,
37.     timeStep int not null,
38.     photoStep varchar(200),
39.     videoStep varchar(200),
40.
41.     primary key(idStep)
42. ) engine=InnoDB;
43.
44. create table Product (
45.     nameProduct varchar(30) not null,
46.     classProduct varchar(30) not null,
47.     unityOfMeasure varchar(10) not null,
48.     subclassProduct varchar(30),
```

```
    photoProduct varchar(200),

    primary key(nameProduct)
) engine=InnoDB;

create table Tool (
    nameTool varchar(30) not null,
    photoTool varchar(200),

    primary key(nameTool)
) engine=InnoDB;

create table Ingredient (
    idRecipe int not null,
    nameProduct varchar(30) not null,
    quantity double not null,
    optional boolean not null default false,

    primary key(idRecipe, nameProduct),
    foreign key(idRecipe) references Recipe(idRecipe),
    foreign key(nameProduct) references Product(nameProduct)
) engine=InnoDB;

create table ProductAvailability (
    username varchar(30) not null,
    nameProduct varchar(30) not null,
    quantity double not null,

    primary key(username, nameProduct),
    foreign key(username) references Account(username),
    foreign key(nameProduct) references Product(nameProduct)
) engine=InnoDB;

create table ToolAvailability (
    username varchar(30) not null,
    nameTool varchar(30) not null,

    primary key(username, nameTool),
    foreign key(username) references Account(username),
    foreign key(nameTool) references Tool(nameTool)
) engine=InnoDB;

create table Rating (
    username varchar(30) not null,
    idRecipe int not null,
    commentRating text,
    rating int not null,
    difficulty int,

    primary key(username, idRecipe),
    foreign key(username) references Account(username),
    foreign key(idRecipe) references Recipe(idRecipe)
) engine=InnoDB;

create table Sequence (
    idRecipe int not null,
    idStep int not null,
    position int not null,

    primary key(idRecipe, idStep),
    foreign key(idRecipe) references Recipe(idRecipe),
    foreign key(idStep) references Step(idStep)
) engine=InnoDB;
```

```
112.
113.create table ToolSet (
114.    idRecipe int not null,
115.    nameTool varchar(30) not null,
116.    optional boolean not null,
117.
118.    primary key(idRecipe, nameTool),
119.    foreign key(idRecipe) references Recipe(idRecipe),
120.    foreign key(nameTool) references Tool(nameTool)
121.) engine=InnoDB;
122.
123.create table Author (
124.    idRecipe int not null,
125.    username varchar(30) not null,
126.
127.    primary key(idRecipe, username),
128.    foreign key(idRecipe) references Recipe(idRecipe),
129.    foreign key(username) references Account(username)
130.) engine=InnoDB;
```

4.2 Codifica delle operazioni

```
1.  DELIMITER $
2.
3.  /* Functions */
4.
5.  create function isStepUnique(description varchar(300)) returns boolean
6.  begin
7.      return (select count(*) from Step where descriptionStep = description) = 0;
8.  end $
9.
10. /* Stored Procedures */
11.
12. drop procedure if exists addAuthor $
13. create procedure addAuthor(in username varchar(30), pass varchar(30), email varchar(30))
14. language sql
15. sql security invoker
16. comment 'Add an author'
17. begin
18.     insert into Account values (username, md5(pass), email, now());
19. end $
20.
21. drop procedure if exists addProduct $
22. create procedure addProduct(in nameProduct varchar(30), classProduct varchar(30), unityOfMeasure varch
    ar(10), subclassProduct varchar(30), photoProduct varchar(200))
23. language sql
24. sql security invoker
25. comment 'Add a product'
26. begin
27.     insert into Product values (nameProduct, classProduct, unityOfMeasure, subclassProduct, photoProdu
    ct);
28. end $
29.
30. drop procedure if exists addTool $
31. create procedure addTool(in nameTool varchar(30), photoTool varchar(200))
32. language sql
33. sql security invoker
34. comment 'Add a tool'
35. begin
36.     insert into Tool values (nameTool, photoTool);
37. end $
```

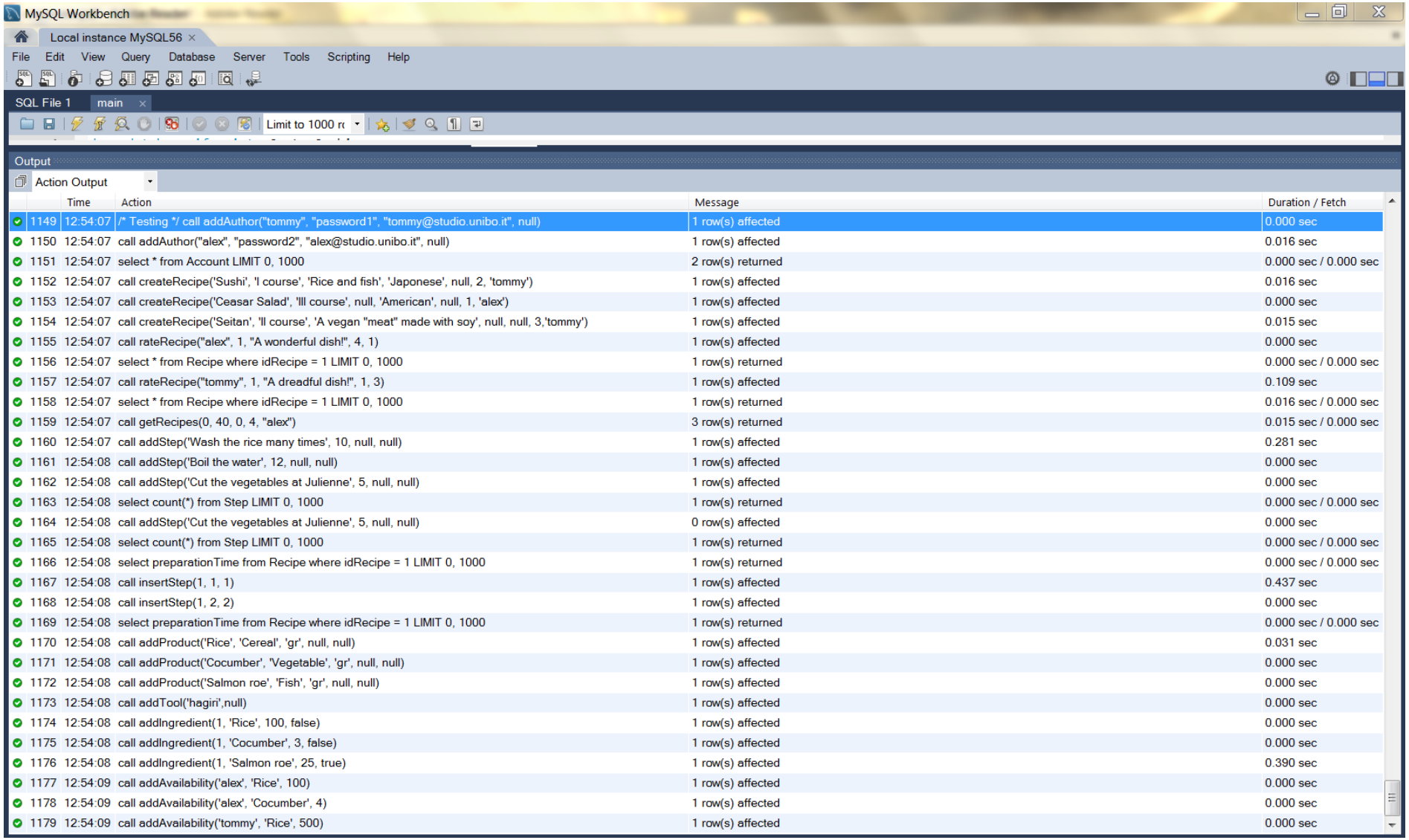
```
38.
39. drop procedure if exists addIngredient $
40. create procedure addIngredient(in idRecipe integer, nameProduct varchar(30), quantity double, optional
     boolean)
41. language sql
42. sql security invoker
43. comment 'Add an ingredient'
44. begin
45.     insert into Ingredient values (idRecipe, nameProduct, quantity, optional);
46. end $
47.
48. drop procedure if exists addAvailability $
49. create procedure addAvailability(in username varchar(30), nameProduct varchar(30), quantity double)
50. language sql
51. sql security invoker
52. comment 'Add an availability'
53. begin
54.     insert into ProductAvailability values (username, nameProduct, quantity);
55. end $
56.
57. drop procedure if exists addStep $
58. create procedure addStep(in descriptionStep varchar(300), timeStep int, photoStep varchar(200), videoS
    tep varchar(200))
59. language sql
60. sql security invoker
61. comment 'Add a step'
62. begin
63.     if isStepUnique(descriptionStep) then
64.         Insert
65.            into Step (descriptionStep, timeStep, photoStep, videoStep)
66.            values (descriptionStep, timeStep, photoStep, videoStep);
67.     end if;
68. end $
69.
70. drop procedure if exists insertStep $
71. create procedure insertStep(in idRecipe integer, idStep integer, position integer)
72. language sql
73. sql security invoker
74. comment 'Insert a step in a sequence'
75. begin
76.     insert into Sequence values (idRecipe, idStep, position);
77. end $
78.
79. drop procedure if exists createRecipe $
80. create procedure createRecipe(in nameRecipe varchar(30), typeRecipe varchar(30), descriptionRecipe tex
    t,
81.     cuisine varchar(30), regionalOrigin varchar(30), difficulty integer, username varchar(30))
82. language sql
83. sql security invoker
84. comment 'Create a recipe'
85. begin
86.     insert into Recipe (nameRecipe, typeRecipe, descriptionRecipe, cuisine, regionalOrigin, preparatio
    nTime, difficultyAuthor, username)
87.         values (nameRecipe, typeRecipe, descriptionRecipe, cuisine, regionalOrigin, 0, difficultyAutho
    r, username);
88. end $
89.
90. drop procedure if exists rateRecipe $
91. create procedure rateRecipe(in username varchar(30), idRecipe int, commentRating text, rating int, dif
    ficulty int)
92. language sql
93. sql security invoker
94. comment 'Rate a recipe'
```

```
95. begin
96.     insert into Rating values (username, idRecipe, commentRating, rating, difficulty);
97. end $
98.
99. drop procedure if exists getRecipes $
100.create procedure getRecipes(in timeMin int, timeMax int, diffMin int, diffMax int, username varchar(3
    0))
101.language sql
102.sql security invoker
103.comment
    'Get the possible recipes for a given user accordingly by time, difficulty and avaiability, ordered by
     rating'
104.begin
105.    select Recipe.*
106.    from Recipe
107.    where
108.        preparationTime between timeMin and timeMax and
109.        (
110.            (difficultyAuthor between diffMin and diffMax) or
111.            (difficultyUsers between diffMin and diffMax)
112.        ) and
113.        idRecipe not in (
114.            select idRecipe
115.            from Ingredient i
116.            where
117.                i.optional = false and
118.                not exists (
119.                    select *
120.                    from ProductAvailability a
121.                    where
122.                        a.username = username and
123.                        i.nameProduct = a.nameProduct and
124.                        i.quantity <= a.quantity))
125.    order by rating desc;
126.end $
127.
128./* Triggers */
129.
130./* Automatically update the overall rating of a recipe. */
131.drop trigger if exists updateRating $
132.create trigger updateRating
133.after insert on Rating
134.for each row
135.begin
136.    declare overallRating double;
137.    declare overallDifficulty double;
138.
139.    select avg(rating) into overallRating
140.    from Rating
141.    where idRecipe = new.idRecipe;
142.
143.    update Recipe
144.    set rating = overallRating
145.    where idRecipe = new.idRecipe;
146.
147.    if (new.difficulty != null) then
148.        select avg(difficulty) into overallDifficulty
149.        from Rating
150.        where idRecipe = new.idRecipe;
151.
152.        update Recipe
153.        set difficultyUsers = overallDifficulty
154.        where idRecipe = new.idRecipe;
```

```
155.     end if;
156. end $
157.
158. /* Automatically update the overall time needed for a recipe. */
159. drop trigger if exists updateTime $
160. create trigger updateTime
161. after insert on Sequence
162. for each row
163. begin
164.     declare overallTime int;
165.
166.     select sum(timeStep) into overallTime
167.     from Sequence natural join Step
168.     where idRecipe = new.idRecipe;
169.
170.     update Recipe
171.     set preparationTime = overallTime
172.     where idRecipe = new.idRecipe;
173. end $
174.
175. DELIMITER;
```

5 Testing

```
1.  call addAuthor("tommy", "password1", "tommy@studio.unibo.it");
2.  call addAuthor("alex", "password2", "alex@studio.unibo.it");
3.
4.  select * from Account;
5.
6.  call createRecipe('Sushi', 'I course', 'Rice and fish', 'Japonese', null, 2, 'tommy');
7.  call createRecipe('Ceasar Salad', 'III course', null, 'American', null, 1, 'alex');
8.  call createRecipe('Seitan', 'II course', 'A vegan "meat" made with soy', null, null, 3,'tommy');
9.
10. call rateRecipe("alex", 1, "A wonderful dish!", 4, 1);
11. select * from Recipe where idRecipe = 1;
12. call rateRecipe("tommy", 1, "A dreadful dish!", 1, 3);
13. select * from Recipe where idRecipe = 1;
14.
15. call getRecipes(0, 40, 0, 4, "alex");
16.
17. call addStep('Wash the rice many times', 10, null, null);
18. call addStep('Boil the water', 12, null, null);
19. call addStep('Cut the vegetables at Julienne', 5, null, null);
20.
21. select preparationTime from Recipe where idRecipe = 1;
22. call insertStep(1, 1, 1);
23. call insertStep(1, 2, 2);
24. select preparationTime from Recipe where idRecipe = 1;
25.
26. call addProduct('Rice', 'Cereal', 'gr', null, null);
27. call addProduct('Cocumber', 'Vegetable', 'gr', null, null);
28. call addProduct('Salmon roe', 'Fish', 'gr', null, null);
29.
30. call addTool('hangiri',null);
31.
32. call addIngredient(1, 'Rice', 100, false);
33. call addIngredient(1, 'Cocumber', 3, false);
34. call addIngredient(1, 'Salmon roe', 25, true);
35.
36. call addAvailability('alex', 'Rice', 100);
37. call addAvailability('alex', 'Cocumber', 4);
38. call addAvailability('tommy', 'Rice', 500);
```



	Time	Action	Message	Duration / Fetch
1149	12:54:07	/* Testing */ call addAuthor("tommy", "password1", "tommy@studio.unibo.it", null)	1 row(s) affected	0.000 sec
1150	12:54:07	call addAuthor("alex", "password2", "alex@studio.unibo.it", null)	1 row(s) affected	0.016 sec
1151	12:54:07	select * from Account LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec
1152	12:54:07	call createRecipe('Sushi', 'I course', 'Rice and fish', 'Japonese', null, 2, 'tommy')	1 row(s) affected	0.016 sec
1153	12:54:07	call createRecipe('Ceasar Salad', 'III course', null, 'American', null, 1, 'alex')	1 row(s) affected	0.000 sec
1154	12:54:07	call createRecipe('Seitan', 'II course', 'A vegan "meat" made with soy', null, null, 3,'tommy')	1 row(s) affected	0.015 sec
1155	12:54:07	call rateRecipe("alex", 1, "A wonderful dish!", 4, 1)	1 row(s) affected	0.000 sec
1156	12:54:07	select * from Recipe where idRecipe = 1 LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
1157	12:54:07	call rateRecipe("tommy", 1, "A dreadful dish!", 1, 3)	1 row(s) affected	0.109 sec
1158	12:54:07	select * from Recipe where idRecipe = 1 LIMIT 0, 1000	1 row(s) returned	0.016 sec / 0.000 sec
1159	12:54:07	call getRecipes(0, 40, 0, 4, "alex")	3 row(s) returned	0.015 sec / 0.000 sec
1160	12:54:07	call addStep('Wash the rice many times', 10, null, null)	1 row(s) affected	0.281 sec
1161	12:54:08	call addStep('Boil the water', 12, null, null)	1 row(s) affected	0.000 sec
1162	12:54:08	call addStep('Cut the vegetables at Julienne', 5, null, null)	1 row(s) affected	0.000 sec
1163	12:54:08	select count(*) from Step LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
1164	12:54:08	call addStep('Cut the vegetables at Julienne', 5, null, null)	0 row(s) affected	0.000 sec
1165	12:54:08	select count(*) from Step LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
1166	12:54:08	select preparationTime from Recipe where idRecipe = 1 LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
1167	12:54:08	call insertStep(1, 1, 1)	1 row(s) affected	0.437 sec
1168	12:54:08	call insertStep(1, 2, 2)	1 row(s) affected	0.000 sec
1169	12:54:08	select preparationTime from Recipe where idRecipe = 1 LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
1170	12:54:08	call addProduct('Rice', 'Cereal', 'gr', null, null)	1 row(s) affected	0.031 sec
1171	12:54:08	call addProduct('Cocumber', 'Vegetable', 'gr', null, null)	1 row(s) affected	0.000 sec
1172	12:54:08	call addProduct('Salmon roe', 'Fish', 'gr', null, null)	1 row(s) affected	0.000 sec
1173	12:54:08	call addTool('hagiri',null)	1 row(s) affected	0.000 sec
1174	12:54:08	call addIngredient(1, 'Rice', 100, false)	1 row(s) affected	0.000 sec
1175	12:54:08	call addIngredient(1, 'Cocumber', 3, false)	1 row(s) affected	0.000 sec
1176	12:54:08	call addIngredient(1, 'Salmon roe', 25, true)	1 row(s) affected	0.390 sec
1177	12:54:09	call addAvailability('alex', 'Rice', 100)	1 row(s) affected	0.000 sec
1178	12:54:09	call addAvailability('alex', 'Cocumber', 4)	1 row(s) affected	0.000 sec
1179	12:54:09	call addAvailability('tommy', 'Rice', 500)	1 row(s) affected	0.000 sec