Database Management Systems, A.Y. 2018/2019
Master Degree in Computer Engineering
Master Degree in Telecommunication Engineering

Homework 2 – Conceptual Design

Deadline: April 3, 2018

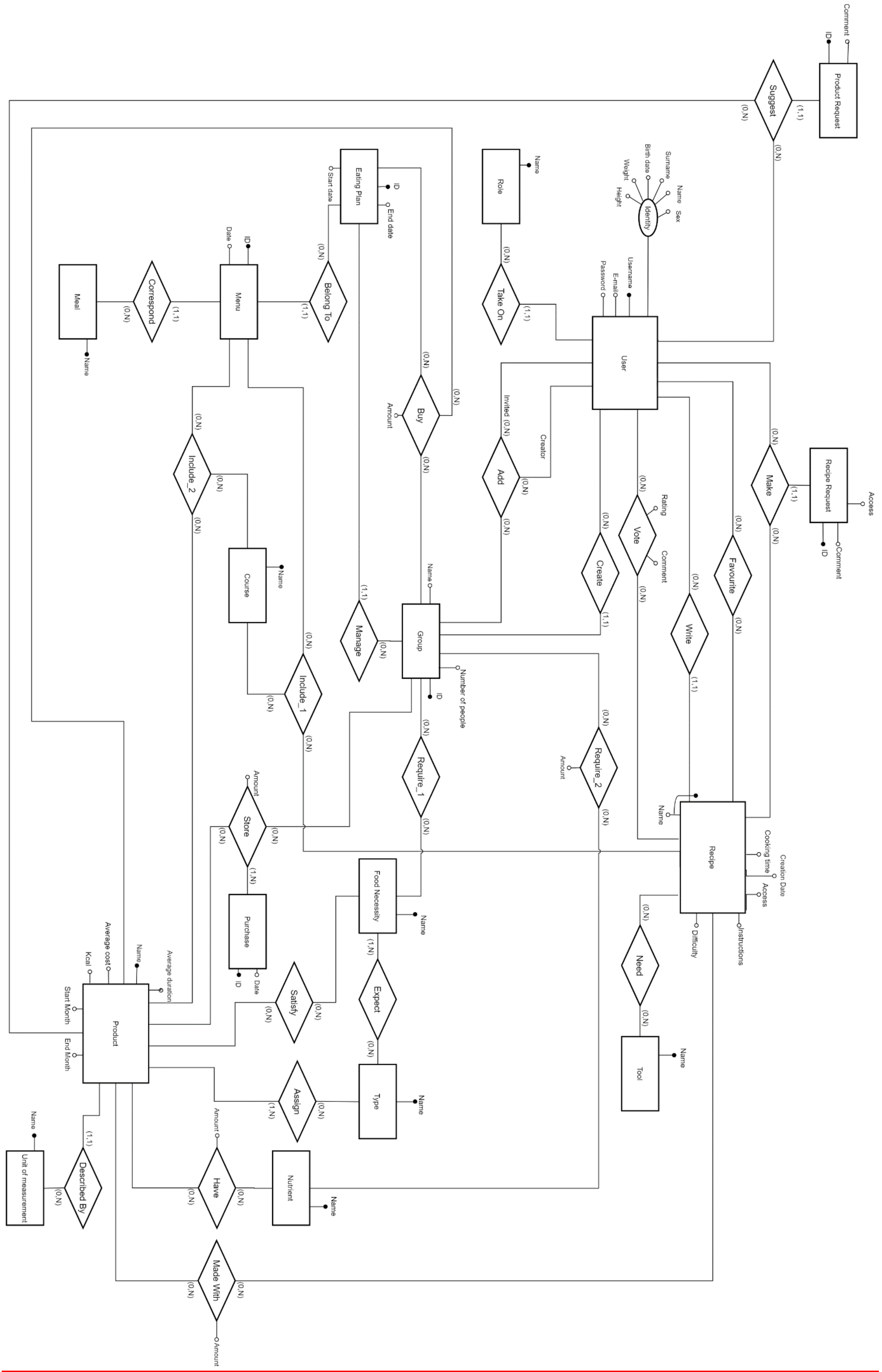
Group Four++	Project AutoChef	
Last Name	First Name	Student Number
Bordignon	Benvenuto	1210175
Di Nardo Di Maio	Raffaele	1204879
Fabris	Cristina	1205722
Perin	Matteo	1205718
Pistilli	Davide Dravindran	1204880

Variations to the Requirement Analysis

At the creation of the subscriber account, the application creates automatically one group that contains only the user himself (it considers the food necessities of the user).

The virtual user was removed as a standalone entity. The creator of a group can change the number of its members and manually add additional food necessities in order to account for people who are not subscribed to AutoChef.

Entity-Relationship Schema



Product Request
ID
Comment
Suggest
(1,1)
(0,N)
(0,N)
User
Identity
Name
Surname
Sex
Birth date
Weight
Height
Username
E-mail
Password
Role
Name
Take On
(1,1)
(0,N)
Eating Plan
ID
Start date
End date
Belong To
(0,N)
(1,1)
Menu
ID
Date
Correspond
(1,1)
(0,N)
Meal
Name
Buy
Amount
(0,N)
(0,N)
(0,N)
Add
Invited (0,N)
Creator
(0,N)
(0,N)
Vote
Rating
Comment
(0,N)
(0,N)
Create
(0,N)
(1,1)
Make
(0,N)
(1,1)
(0,N)
Recipe Request
ID
Comment
Access
Favourite
(0,N)
(0,N)
Write
(0,N)
(1,1)
Include_2
(0,N)
(0,N)
(0,N)
Course
Name
Manage
(1,1)
(0,N)
Group
Name
ID
Number of people
Include_1
(0,N)
(0,N)
(0,N)
Require_1
(0,N)
(0,N)
Require_2
Amount
(0,N)
(0,N)
Recipe
Name
Cooking time
Creation Date
Access
Difficulty
Instructions
Store
Amount
(0,N)
(0,N)
(1,N)
Purchase
ID
Date
Food Necessity
Name
Expect
(1,N)
(0,N)
Satisfy
(0,N)
(0,N)
Need
(0,N)
(0,N)
Tool
Name
Product
Name
Average cost
Kcal
Average duration
Start Month
End Month
Assign
(1,N)
(0,N)
Type
Name
Described By
(1,1)
(0,N)
Unit of measurement
Name
Have
Amount
(0,N)
(0,N)
Nutrient
Name
Made With
Amount
(0,N)
(0,N)

Data Dictionary

Entities Table

Entity	Description	Attributes	Identifier
Course	Course during a meal (Starter, First Course, Second Course, Side, Dessert).	• Name, name of the course, text.	Name
Eating Plan	Ordered sequence of menus organized for a certain time period.	• End date, date of the day in which the plan ends, date. • ID, identifier of the eating plan, integer. • Start date, date of the day in which the plan begins, date.	ID
Food Necessity	Allergy or food preference.	• Name, name of the allergy or the food preference, text.	Name
Group	Set of one or more users.	• ID, identifier of the group, integer. • Name, name of the group, text. • Number of people, number of participants of the group, integer.	ID
Meal	A meal during a day (e.g. breakfast, lunch, ...).	• Name, name that identifies the meal, text.	Name
Menu	Set of dishes that are served during a meal.	• Date, day in which the menu will be prepared, date. • ID, identifier of the menu, integer.	ID
Nutrient	Basic nutrient (e. g. protein, carbohydrate, …).	• Name, name of the nutrient, text.	Name
Product	Raw material or processed food.	• Average cost, average cost of the product, real. • Average duration, average expiration time of the product, integer. • End month, last month in which the product is available, integer. • Kcal, amount of kcal of the product, real. • Name, name of the product, text. • Start month, first month in which the product is available, integer.	Name
Product Request	Request to add new product in the database.	• Comment, description of the product that a user wants to add, text. • ID, identifier of the product request, integer.	ID
Purchase	Purchase of a set of products.	• Date, date in which the purchase is made, date.	ID

		• ID, identifier of the purchase, integer.	
Recipe	Set of instructions that users can follow to transform a set of products into courses.	• Access, current state of the recipe (private or public), text. • Cooking time, minutes needed to complete the recipe, integer. • Creation Date, date in which the author created the recipe, date. • Difficulty, difficulty of the recipe (from 1 to 5), integer. • Instructions, steps to follow in order to prepare the recipe, text. • Name, name of the recipe, text. • Username, username of the author, from the entity User, text.	Name Username (from the entity User)
Recipe Request	Request to make a recipe public or to suggest some modifications to it.	• Access, current state of the recipe (private or public), text. • Comment, details of the request, text. • ID, identifier of the recipe request, integer.	ID
Role	Role that a user can perform inside the database.	• Name, name that identifies the role of a user (Subscriber, Moderator, Banned), text.	Name
Tool	Instrument that has to be used in the preparation of a recipe (e. g. oven, spoon, …).	• Name, name of the instrument, text.	Name
Type	Type of food (e. g. vegetable, meat, dairy product, …).	• Name, name of the type of food, text.	Name
Unit of measurement	Unit of measurement of a product.	• Name, name of the unit of measurement, text.	Name
User	User of AutoChef.	• E-mail, e-mail of the user, text. • Identity, set of personal information that a user needs to specify during the subscription 1. Birth date, birthday of the user, date. 2. Height, height of the user, real. 3. Name, name of the user, text. 4. Sex, sex of the user, text. 5. Surname, surname of the user, text. 6. Weight, weight of the user, real. • Password, personal password for the authentication of the user, text. • Username, unique name that	Username

		identifies a user in the database, text.	

Relationship Table

Relationship	Description	Component Entities	Attributes
Add	Addition of a User, called Invited, to a group created by a User called Creator.	• Group, (0, N). • User (Creator), (0, N). • User (Invited), (0, N).	
Assign	Associates a Type of food to a Product.	• Product, (1, N). • Type, (0, N).	
Belong to	Associates a set of Menus to an Eating Plan.	• Eating Plan, (0, N). • Menu, (1, 1).	
Buy	Indicates which Products a Group needs to buy for a specific Eating Plan.	• Eating plan, (0, N). • Group, (0, N). • Product, (0, N).	• Amount, quantity of a product that a group needs to buy, real.
Correspond	Associates a Menu to the Meal (Breakfast, Lunch, …) in which it will be consumed.	• Meal, (0, N). • Menu, (1,1).	
Create	Creation of a Group by a User.	• Group, (1, 1). • User, (0, N).	
Described by	Associates a Product to its Unit of measurement.	• Product, (1,1). • Unit of Measurement, (0, N).	
Expect	Associates a set of Types of food that can be consumed with a Food Necessity.	• Food Necessity, (1, N). • Type, (0, N).	
Favourite	Allows a User to mark their favourite public Recipes.	• Recipe, (0, N). • User, (0, N).	
Have	Associates a Product to the set of Nutrients it contains.	• Nutrient, (0, N). • Product, (0, N).	• Amount, quantity of a nutrient in a product, real.
Include_1	Addition of a Recipe to a specific Course in a Menu.	• Course, (0, N). • Menu, (0, N). • Recipe, (0, N).	
Include_2	Addition of a Product to a specific Course in a Menu.	• Course, (0, N). • Menu, (0, N). • Product, (0, N).	
Made with	Associates a Recipe to the Products it needs.	• Product, (0, N). • Recipe, (0, N).	• Amount, quantity of a product needed for the recipe, real.
Make	Creates request of improvements or publication of a recipe.	• Recipe, (0, N). • Recipe Request (1,	

		1). ● User (0, N).	
Manage	Association of an Eating Plan to a Group.	● Eating Plan, (1, 1). ● Group, (0, N).	
Need	Associates a Recipe to the set of Tools needed to prepare it.	● Recipe, (0, N). ● Tool, (0, N).	
Require_1	Associates a Group to a set of Food Necessities (allergies or food preferences).	● Food Necessity, (0, N). ● Group, (0, N).	
Require_2	Associates the Group to the amount of the Nutrient needed.	● Group, (0, N). ● Nutrient, (0, N).	● Amount, quantity of nutrient needed by a group, real.
Satisfy	Associates a Product to the Food Necessities that it satisfies.	● Food Necessity, (0, N). ● Product, (0, N).	
Store	Stores a Product, bought by a Group, after its Purchase.	● Group, (0, N). ● Product, (0, N). ● Purchase, (1, N).	● Amount, quantity of the product stored after a purchase, real.
Suggest	Proposal to add a product to the database.	● Product, (0, N). ● Product Request, (1, 1). ● User, (0, N).	
Take on	Associates each User to a Role, that defines its permissions.	● Role, (0, N). ● User, (1,1).	
Vote	Addition of a review by a User to a public Recipe.	● Recipe, (0, N). ● User, (0, N).	● Comment, a short review of the recipe, text. ● Rating, number of stars (1-5), integer.
Write	Creation of a Recipe by a User.	● Recipe, (1, 1). ● User, (0, N).	

External Constraints

1. A group cannot have two overlapping eating plans (eg. the start date of a second plan cannot be between the start and end dates of the first plan).
2. Since only the username is used as an identifier for the subscriber, it must be checked that emails used are unique as well.
3. When a user registers to AutoChef, the application automatically creates a personal group in the database. It is transparent to the user and it contains only the new subscriber. This allows the application to handle all of its functions through groups instead of both groups and specific users. Furthermore, when a user creates a group, they are added to it automatically by the application.
4. A recipe request identifies a request by a user regarding a recipe. There are two types of request: a publication request and a suggestion request. The former allows the recipe's author to share their creation with the community. The latter allows any user to propose

changes to already existing public recipes. The access attribute identifies the current visibility state of the recipe: if it is set to private it marks a publication request (because a private recipe can only be seen by its author). Otherwise it marks a suggestion request.
5. A product request represents the request of a user to add a product to the product list. All products are visible to all users, who can add them to their plans or use them as ingredients for their own recipes.
6. Recipe and product request are only visible to moderators.
7. The application should check the affinity between food necessities and the types of products used to compose a meal and tell the user what should be changed to respect all the food necessities of the group.
8. Rating and Difficulty of a recipe must be in the range [1, 5].
9. Each user has only access to the eating plans linked to the groups they are a member of.
10. If a recipe is private the user that has written it can modify or delete it at any time . When a recipe becomes public, its author can no longer modify it directly, but they must make a request to do so.
11. Each user has only access to groups he is a member of.
12. Each recipe has a rating, equal to the average of ratings made by users, that is computed by the application.
13. A user can only favourite recipes that he did not write in order to avoid data duplication. The personal recipe book unifies the saved recipes and those written by the user.

Functional Requirements Satisfaction Check

The DBMS has the following functional requirements:

- **Manage two account types: users and moderators**: the binary relationship “Take On” between the entity “User” and the entity “Role” allows the database to distinguish between normal users and moderators.
- **Store user data: name, surname, date of birth, username, password, e-mail, allergies, food preferences, weight, height, nutrient needs:** each user must register to the system and each relevant information is saved in the database through the composite attribute “Identity” of the entity “User”. Nutrient needs are evaluated by the system for each group after every update (by means of specific formulas). These are linked to the “Group” entity through the relationship “Require_2” that has the “Amount” as an attribute. The entity “Food Necessity” (that includes food preferences and allergies) is linked with the entity “Group” through the relationship “Require_1” with cardinality (0,N) and the entity “Group” is linked with the entity “User” through two relationships: “Create” with cardinality (1,1) and “Add” with cardinality (0,N).
- **Store recipe information: name, author, creation date, difficulty, tools, product list, course, allergens, cost, rating, suitable food preferences, preparation time, procedure, season**: the entity “Recipe” stores the name, creation date, difficulty, access, instructions, cooking time for each instance of the recipe. The relationship “Write” associates each recipe to one and only one (1,1) user. The relationship “Need” associates a recipe to zero or more (0,N) Tools. The attribute “Rating” of the binary relationship “Vote” between the entity “User” and the entity “Recipe” allows to save user ratings for every recipe. The entity “Recipe” is linked to the entity “Product” to arrange the list of products needed for a certain recipe. Furthermore, the entity “Product” is connected through a binary relationship “Satisfy” to the entity “Food Necessity” so at the application level it is possible to understand the food preferences and the allergens of a recipe by knowing the products it is made of.
- **Manage a personal recipe book, with the possibility to add existing recipes, create new ones and remove those already present:** The personal recipe book is represented by two relationships: “Write” and “Favourite”. “Write” associates each “User” to each

"Recipe" he/she wrote. "Favourite", on the other hand, allows a "User" to "Favourite" other public recipes from the database.

- **Allow users to share their created recipes with the community after confirmation from a moderator:** The entity user can make a request to share a personal recipe through the ternary relationship "Make". At the application level a moderator see the request and can approve or reject it. When a recipe becomes public its author can no longer modify it directly but he can only make a recipe request.
- **Manage the creation and deletion of user groups:** the entity "User" is linked to the entity "Group" with two different relationships: the binary one ("Create") allows a user to create many groups (0,N) and at the same time allows a group to be created by only one user (1,1) whereas the ternary relationship "Add" allows a "User" (the "Group" Creator) to add another "User" (called Invited in the schema) to a "Group" he already created. The deletion of groups is managed by the application.
- **Allow users to create fake accounts identified only by: name, allergies, food preferences, nutrient needs:** This function has been deleted. Instead it is either possible to add a new user to a group (so they must register to AutoChef) or to increase the attribute "Number of people" in the entity "Group" in order to take into account additional members who are not registered. The nutrient needs are averaged across all members of a "Group".
- **Store schedule data: start date, duration, menu list, number of people, user list:** the entity "Eating Plan" stores start date and end date for each instance of the eating plan. The binary relationship between the entity "Eating Plan" and "Menu" allows an "Eating Plan" to be associated to more menus. Finally the cardinality (1,1) of the entity "Eating plan" in the relationship "Manage" ensures that each "Eating Plan" is related to one and only one "Group" thus allowing to know the number of people that use the eating plan.
- **Manage the creation by users of an eating plan, both for themselves and for a group:** the binary relationship "Manage" between the entity "Group" and "Eating plan" allows a group (that can be composed even by one person) to add/modify or delete an eating plan.
- **Assist users during plan creation by considering their personal settings: personal recipe book, allergies, food preferences, nutrient needs, pantry, group members, seasonal products and average expiration times:**
This task is done at the application level. "Recipe" suggestions are taken from the recipe book of the "User" that is editing the "Eating Plan". Food necessities and nutrient needs are considered The data needed for this task can be found by looking at all relationships connected to "Group" with the entity food necessity, with the relation store (which determines the content of the pantry) and with the products attributes.
- **Allow users to modify and delete their plans at any time:** the binary relationship between the entity "Group" and "Eating plan" allow a group (that can be composed even by one person) to add/modify or delete an eating plan. The relationship between the entity "Group" and the entity "User" has been already explained
- **Create a shopping list based on the plan and on the remaining items in the pantry:** at the application level the system suggests the quantity to buy using the information amount in the ternary relationship "Store" between the entity "Group", "Purchase", "Product" and saves it by means of the relationship "Buy" which can be used to determine the shopping list.
- **Compute the average cost for the shopping list. If it is a shared shopping list divide this cost between all users in the group:** this function is applied in the application level where the cost for each user is calculated dividing the cost of the shopping list by the number of people in the group.
- **Manage adding and removing items from the pantry**: the ternary relationship "Store" between the entity "Group", "Purchase" and "Product" allows the group (that can be composed also by only one user) to store or delete each product. In particular the entity "Purchase" keep track of the instance in which a certain element is added or deleted.

- **Allow users to rate recipes:** the binary relationship “Vote” between the entity “User” and the entity “Recipe” with attributes “Rating” and “Comment” and with cardinality (0,N) in both part allows a user to rate and comment different recipes.
- **Allow users to modify their settings at any time:** the entity “User” stores username and password for each user, thus allowing users to log in their respectively account and change their settings whenever they want to.
- **Allow users to add public products, after confirmation from a moderator:** a normal user can through the ternary relationship “Suggest” make a request of adding a new product in the database that is not already present. At application level the moderator can see the request and add or reject the product suggested by the user.
- **Manage user permissions: users can read both public and their personal recipes but can only edit personal ones. They can only access the groups they are in and modify the ones they created. They can read the product list and read/modify their personal storage:** The ability given to the normal user to modify only private is allowed by the system through the attribute access in the entity “Recipe”. In fact at the application level the system manage the recipes as described in the external constraints 5 and 12. Also the fact that every user can only modify his group is described in the external constraint 13. In addition the ternary relationship “Store” allows the group to manage at any time the product that they have purchased.
- **Manage moderator permissions: they can access and edit all public recipes and products. They can see recipe sharing and suggestion proposals and accept/reject them**: at application level the moderator has in general more function than a normal user. In particular he has a control panel where he can see the request of the users and where he can edit all the public recipes included in the database.
- **Allow user suggestions on recipes. These must be confirmed by a moderator.** The entity “User” through the relation with the entity “Recipe Request” can make a request to modify a “Recipes”. The moderator that is user with a role of moderator can, at application level, implement or reject the different suggestions.