UNIVERSITÀ DI PISA

Computer Engineering

Large Scale and Multi-Structured Databases

WORKGROUP TASK 3

Restaurant recommendations using a graph database

STUDENTS:
Cima Lorenzo
Ferrante Nicola
Pampaloni Simone
Scatena Niccolò

Academic Year 2019/2020

Abstract

This project has been developed as a workgroup during the classes of Large Scale and Multi-Structured Databases. The aim of this workgroup is to give a demonstration of the students' capabilities to handle the development of a simple Java application connecting to a graph database, starting from requirements and arriving to implementation.

Contents

1 | Application's modules

In this chapter we present briefly the modules that compose our application. The application is Client-Server, the client has a graphical user interface and the server is a stateful multi-threaded server which use a Neo4j database for data persistency.

1.1 Client

The client application has been developed using JavaFX for the graphical user interface. The Views used are organized in Panels, which groups functionalities for searching Users and Restaurant, a form for User Preferences and Restaurants creation and modification. Furthermore the Users with administrator functionalities will have an administration panel where it is possible to add Cities and Cuisines. Further information on the Client module can be found in Chapter 3 "User Manual".

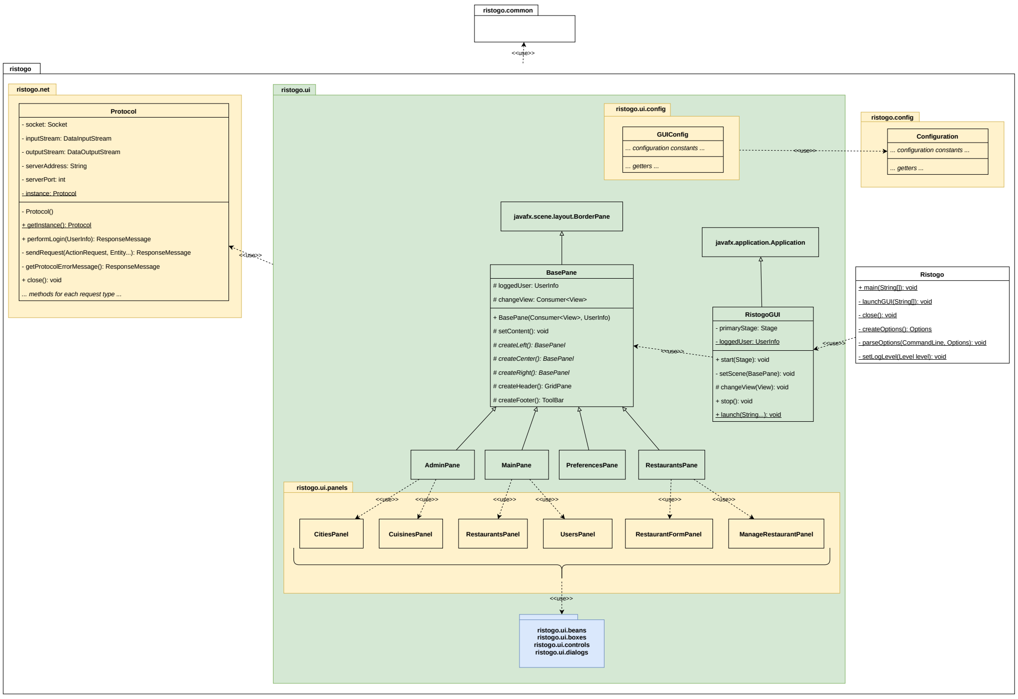

ristogo.common
<<use>>
ristogo
ristogo.net
Protocol
- socket: Socket
- inputStream: DataInputStream
- outputStream: DataOutputStream
- serverAddress: String
- serverPort: int
- instance: Protocol
- Protocol()
+ getInstance(): Protocol
+ performLogin(UserInfo): ResponseMessage
- sendRequest(ActionRequest, Entity...): ResponseMessage
- getProtocolErrorMessage(): ResponseMessage
+ close(): void
... methods for each request type ...
<<use>>
ristogo.ui
ristogo.ui.config
GUIConfig
... configuration constants ...
... getters ...
<<use>>
ristogo.config
Configuration
... configuration constants ...
... getters ...
javafx.scene.layout.BorderPane
javafx.application.Application
BasePane
# loggedUser: UserInfo
# changeView: Consumer<View>
+ BasePane(Consumer<View>, UserInfo)
# setContent(): void
# createLeft(): BasePanel
# createCenter(): BasePanel
# createRight(): BasePanel
# createHeader(): GridPane
# createFooter(): ToolBar
<<use>>
RistogoGUI
- primaryStage: Stage
- loggedUser: UserInfo
+ start(Stage): void
- setScene(BasePane): void
# changeView(View): void
+ stop(): void
+ launch(String...): void
<<use>>
Ristogo
+ main(String[]): void
- launchGUI(String[]): void
- close(): void
- createOptions(): Options
- parseOptions(CommandLine, Options): void
- setLogLevel(Level level): void
AdminPane
MainPane
PreferencesPane
RestaurantsPane
ristogo.ui.panels
<<use>>
<<use>>
<<use>>
<<use>>
<<use>>
<<use>>
CitiesPanel
CuisinesPanel
RestaurantsPanel
UsersPanel
RestaurantFormPanel
ManageRestaurantPanel
<<use>>
ristogo.ui.beans
ristogo.ui.boxes
ristogo.ui.controls
ristogo.ui.dialogs

Figure 1.1: Client UML class diagram.

1.2 Server

The server is a multi-threaded stateful server. The multithreading has been achieved using a reusable thread pool.

For each user connected the state is maintained on the server for all the connection time.

The connection with the database use bolt protocol, a Neo4j specific protocol, which being native is faster and lighter then HTTP based connection. The Data Access Layer has been designed using a Object Graph Mapping library, thus allowing us to easily perform CRUD operation on the DB and to implement complex functionalities using Cypher queries.

To avoid memory exhausting problems the Driver for the database use a pool of sessions that can be reused among different threads, this has some consequences from the point of view of the consistency between memory and database, since some operations may be deferred or done using cached data. Thus to avoid inconsistency problem every time a request is finished we clear and close the session so that the driver can recreate a new one, so that the memory can be freed from cached data and loaded at the next request. In this way we can be sure that the data on which we work is completely consistent with the database and that we have in memory only the data needed for the operation.

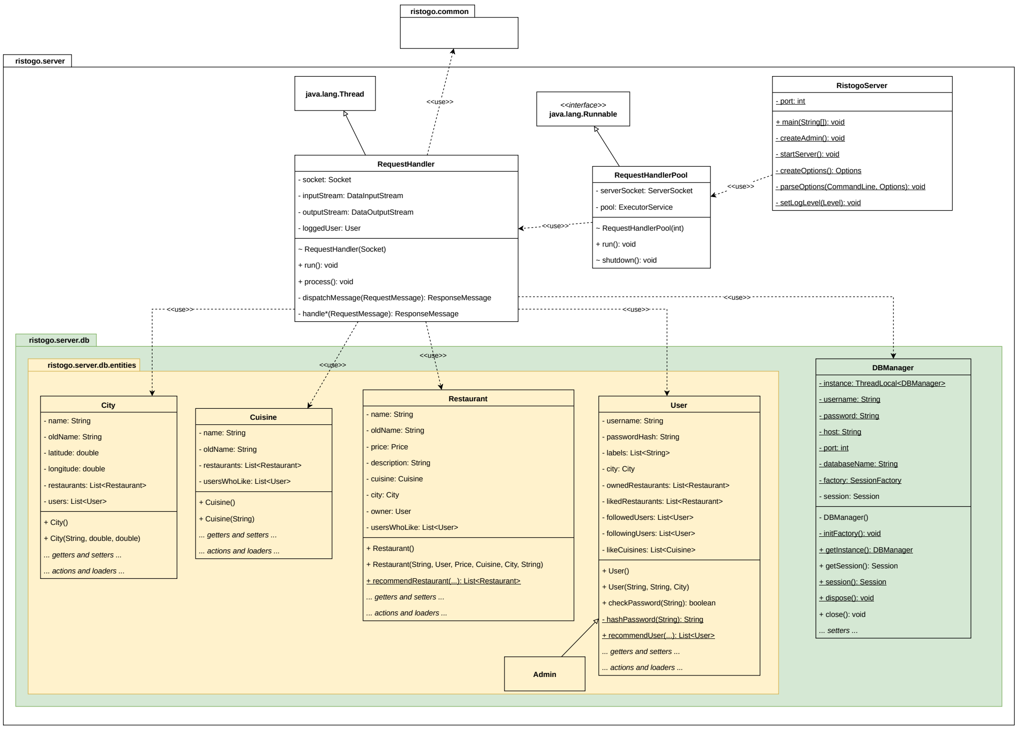

ristogo.common
ristogo.server
java.lang.Thread
<<use>>
<<interface>>
java.lang.Runnable
RistogoServer
- port: int
+ main(String[]): void
- createAdmin(): void
- startServer(): void
- createOptions(): Options
- parseOptions(CommandLine, Options): void
- setLogLevel(Level): void
RequestHandler
- socket: Socket
- inputStream: DataInputStream
- outputStream: DataOutputStream
- loggedUser: User
~ RequestHandler(Socket)
+ run(): void
+ process(): void
- dispatchMessage(RequestMessage): ResponseMessage
- handle*(RequestMessage): ResponseMessage
RequestHandlerPool
- serverSocket: ServerSocket
- pool: ExecutorService
~ RequestHandlerPool(int)
+ run(): void
~ shutdown(): void
<<use>>
<<use>>
<<use>>
<<use>>
<<use>>
<<use>>
<<use>>
ristogo.server.db
ristogo.server.db.entities
City
- name: String
- oldName: String
- latitude: double
- longitude: double
- restaurants: List<Restaurant>
- users: List<User>
+ City()
+ City(String, double, double)
... getters and setters ...
... actions and loaders ...
Cuisine
- name: String
- oldName: String
- restaurants: List<Restaurant>
- usersWhoLike: List<User>
+ Cuisine()
+ Cuisine(String)
... getters and setters ...
... actions and loaders ...
Restaurant
- name: String
- oldName: String
- price: Price
- description: String
- cuisine: Cuisine
- city: City
- owner: User
- usersWhoLike: List<User>
+ Restaurant()
+ Restaurant(String, User, Price, Cuisine, City, String)
+ recommendRestaurant(...): List<Restaurant>
... getters and setters ...
... actions and loaders ...
User
- username: String
- passwordHash: String
- labels: List<String>
- city: City
- ownedRestaurants: List<Restaurant>
- likedRestaurants: List<Restaurant>
- followedUsers: List<User>
- followingUsers: List<User>
- likeCuisines: List<Cuisine>
+ User()
+ User(String, String, City)
+ checkPassword(String): boolean
- hashPassword(String): String
+ recommendUser(...): List<User>
... getters and setters ...
... actions and loaders ...
Admin
DBManager
- instance: ThreadLocal<DBManager>
- username: String
- password: String
- host: String
- port: int
- databaseName: String
- factory: SessionFactory
- session: Session
- DBManager()
- initFactory(): void
+ getInstance(): DBManager
+ getSession(): Session
+ session(): Session
+ dispose(): void
+ close(): void
... setters ...

Figure 1.2: Server UML class diagram.

1.3 Common classes

The common classes are serializable classes that are shared between the application modules. These classes are used for communication purposes.

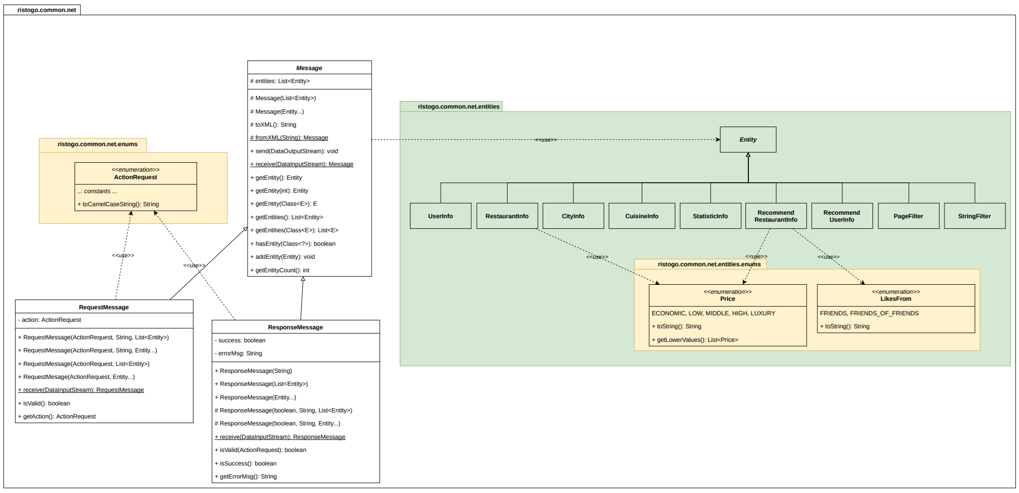

ristogo.common.net
ristogo.common.net.enums
<<enumeration>>
ActionRequest
... constants ...
+ toCamelCaseString(): String
Message
# entities: List<Entity>
# Message(List<Entity>)
# Message(Entity...)
# toXML(): String
# fromXML(String): Message
+ send(DataOutputStream): void
+ receive(DataInputStream): Message
+ getEntity(): Entity
+ getEntity(int): Entity
+ getEntity(Class<E>): E
+ getEntities(): List<Entity>
+ getEntities(Class<E>): List<E>
+ hasEntity(Class<?>): boolean
+ addEntity(Entity): void
+ getEntityCount(): int
<<use>>
<<use>>
<<use>>
RequestMessage
- action: ActionRequest
+ RequestMessage(ActionRequest, String, List<Entity>)
+ RequestMessage(ActionRequest, String, Entity...)
+ RequestMessage(ActionRequest, List<Entity>)
+ RequestMesage(ActionRequest, Entity...)
+ receive(DataInputStream): RequestMessage
+ isValid(): boolean
+ getAction(): ActionRequest
ResponseMessage
- success: boolean
- errorMsg: String
+ ResponseMessage(String)
+ ResponseMessage(List<Entity>)
+ ResponseMessage(Entity...)
# ResponseMessage(boolean, String, List<Entity>)
# ResponseMessage(boolean, String, Entity...)
+ receive(DataInputStream): ResponseMessage
+ isValid(ActionRequest): boolean
+ isSuccess(): boolean
+ getErrorMsg(): String
ristogo.common.net.entities
Entity
UserInfo
RestaurantInfo
CityInfo
CuisineInfo
StatisticInfo
Recommend RestaurantInfo
Recommend UserInfo
PageFilter
StringFilter
<<use>>
<<use>>
<<use>>
ristogo.common.net.entities.enums
<<enumeration>>
Price
ECONOMIC, LOW, MIDDLE, HIGH, LUXURY
+ toString(): String
+ getLowerValues(): List<Price>
<<enumeration>>
LikesFrom
FRIENDS, FRIENDS_OF_FRIENDS
+ toString(): String

Figure 1.3: Server UML class diagram.

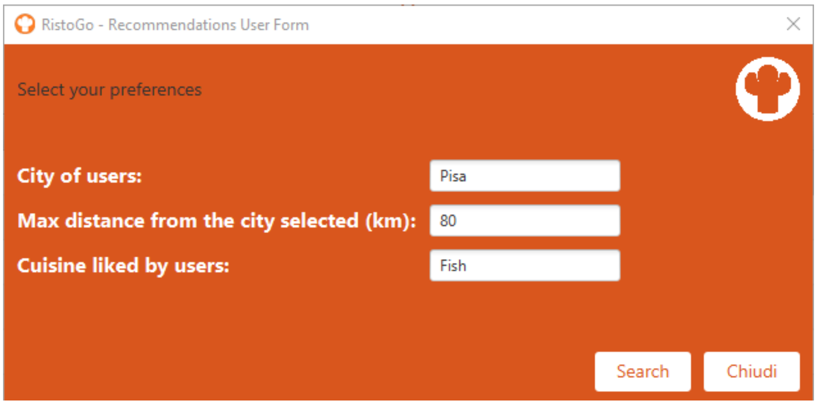
2 | Performed tests

We have tested the application in all of its functions. In particular, all the on-graph queries have been tested multiple times.

Tests have been performed on a test database created from us, whose dump can be found in `db/ristogo.dump`.

2.1 User Recommendations

The user must fill the form shown in Figure 2.1 in the application to perform this query.



RistoGo - Recommendations User Form

Select your preferences

City of users:	Pisa
Max distance from the city selected (km):	80
Cuisine liked by users:	Fish

Figure 2.1: User Recommendations Application Form.

In this test, the query should find users who are not followed by the current user (“simone”), who live in Pisa or live 80 kilometers away from Pisa, who likes a Fish cuisine. The resulting users should be sorted by distance from Pisa.

Figure 2.2 shows the registered users who like Fish cuisine and the cities in which they live.

Figure 2.3 shows the users followed by the current user (named “simone”).

So, as you can see from the table in Figure 2.4, the application returns the correct result as it shows “frank” and “heidi” users who are the users who like the Fish cuisine and who live within a radius of 80 kilometers from Pisa. Note that in the table, as expected, they are sorted by distance from Pisa.

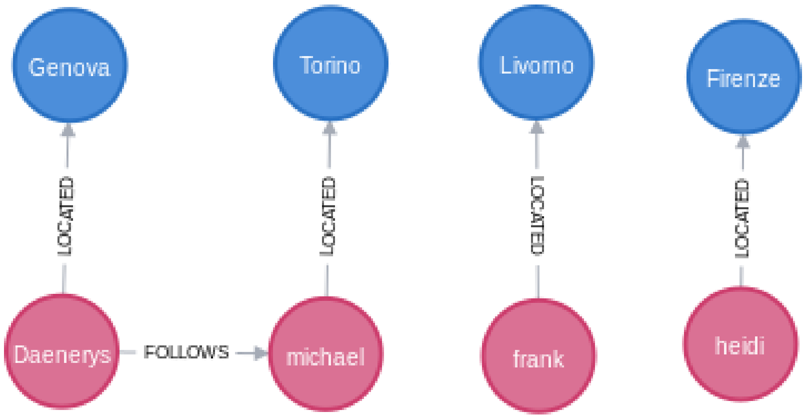


Figure 2.2: Users who like Fish and their cities.

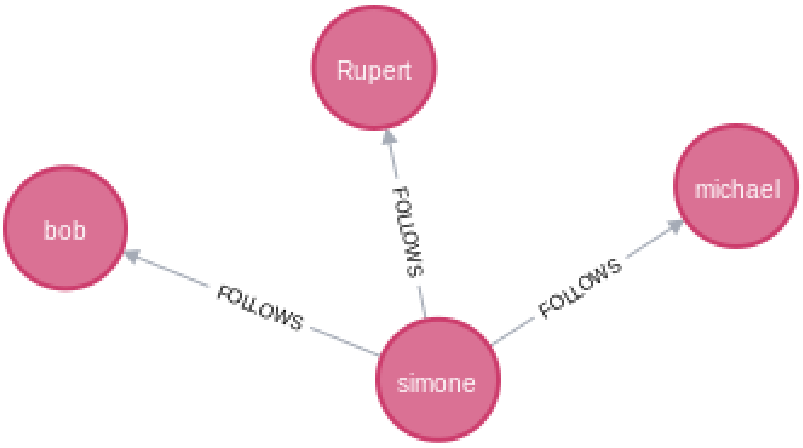


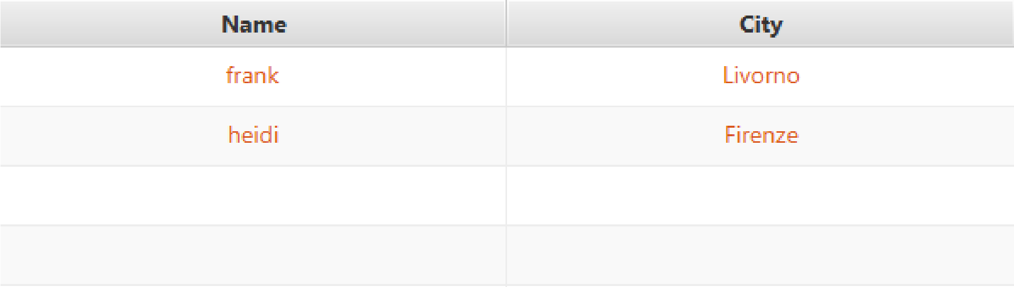
Figure 2.3: Users followed by simone.

2.2 Restaurant Recommendations

We have tested the recommendation on restaurants, here we are providing an example of the correctness of our results.

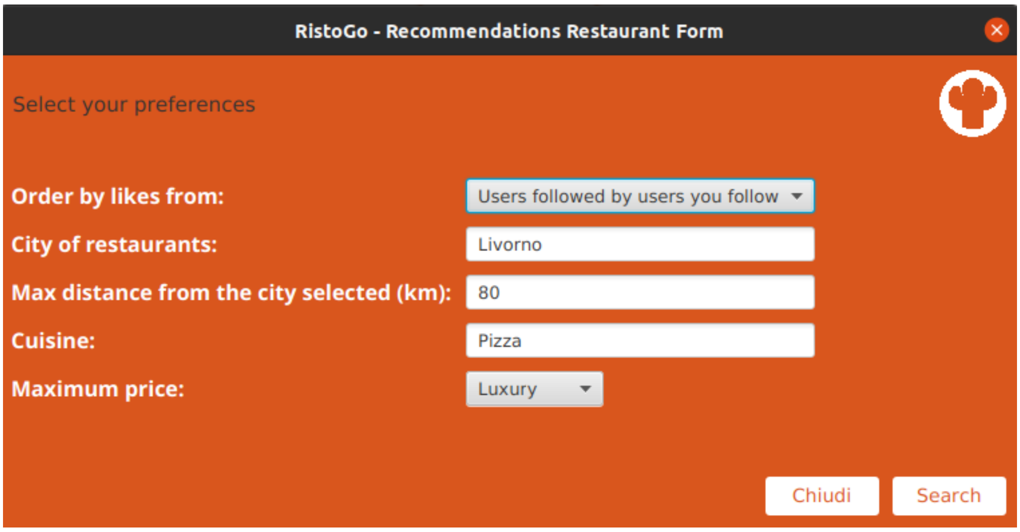
Let's suppose to have the situation shown in Figure 2.5, looking at the user "carol" and his followers/followed users.

Now, suppose that carol want to find a restaurant that `SERVES` Pizza with a price of `Luxury` or lower, at most 80 km distant from Livorno. She would fill the form like in resrecform.



Name	City
frank	Livorno
heidi	Firenze

Figure 2.4: Recommended users by the application.



RistoGo - Recommendations Restaurant Form

Select your preferences

Order by likes from:	Users followed by users you follow
City of restaurants:	Livorno
Max distance from the city selected (km):	80
Cuisine:	Pizza
Maximum price:	Luxury

Figure 2.6: Restaurant Recommendations Form.

In the system we have the Restaurants shown in Figure 2.7 which serves Pizza.

Now the system will suggest to carol the restaurants which serves pizza ordered by the number of likes of friends and friends of friends and excluding the restaurants she already likes or owns.

The result provided by the application is shown in Figure 2.8.

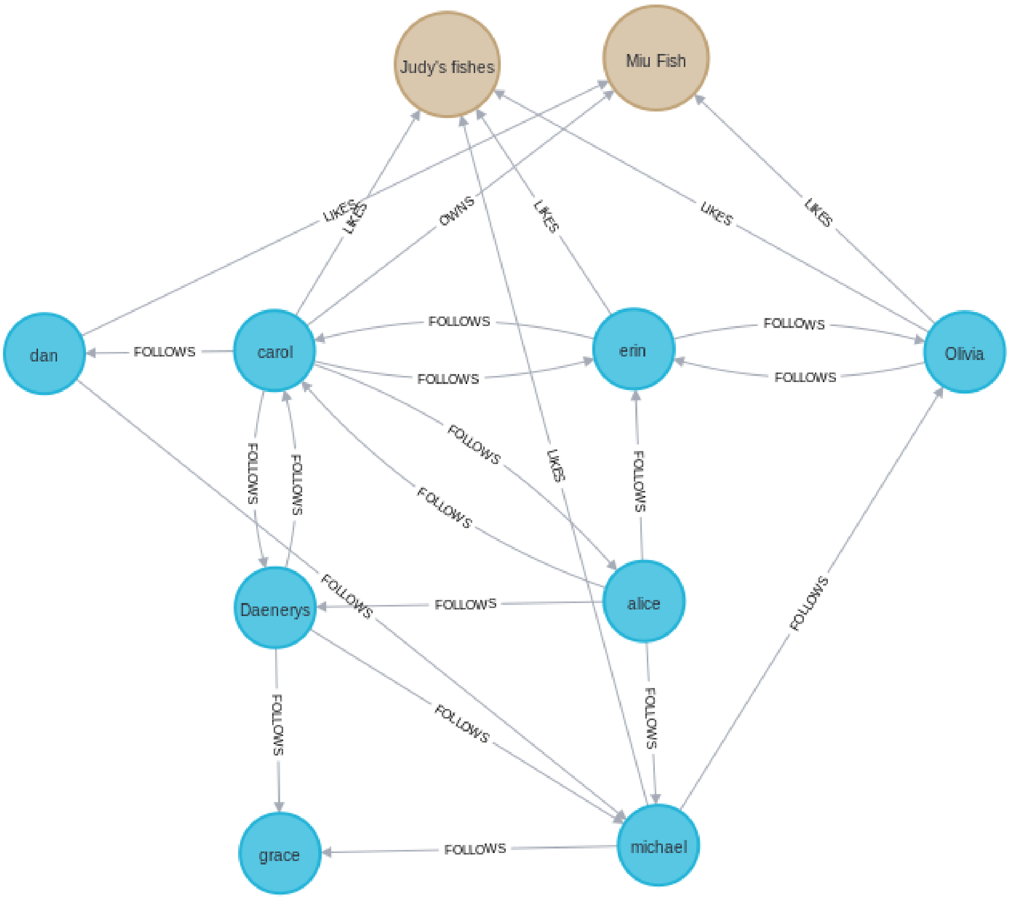
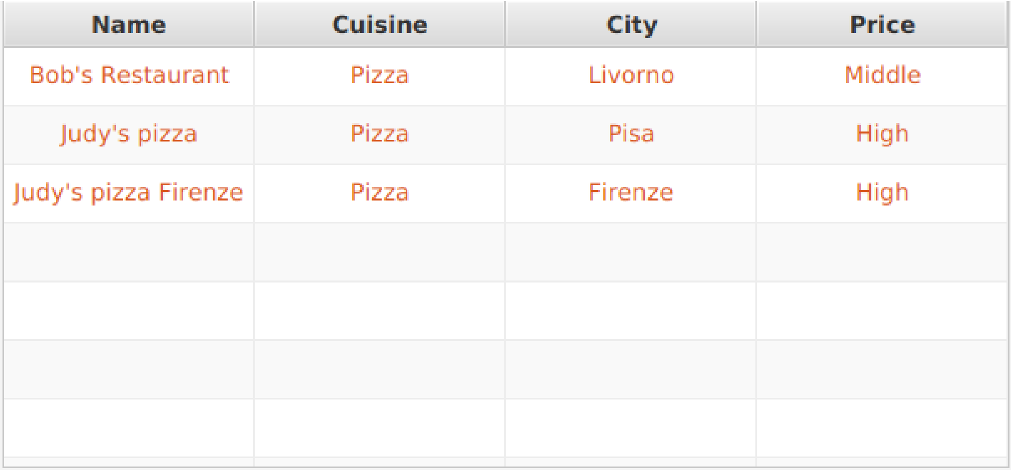


Figure 2.5: Followers of carol and users followed by carol.



Name	Cuisine	City	Price
Bob's Restaurant	Pizza	Livorno	Middle
Judy's pizza	Pizza	Pisa	High
Judy's pizza Firenze	Pizza	Firenze	High

Figure 2.8: Restaurant Recommendations Form.

We can then check in the database that result is correct, in fact looking at the database we have the situation shown in Figure 2.9.

In the figure are shown only the relationships from friends or friends of friends of carol with the restaurants which serves pizza. Counting the

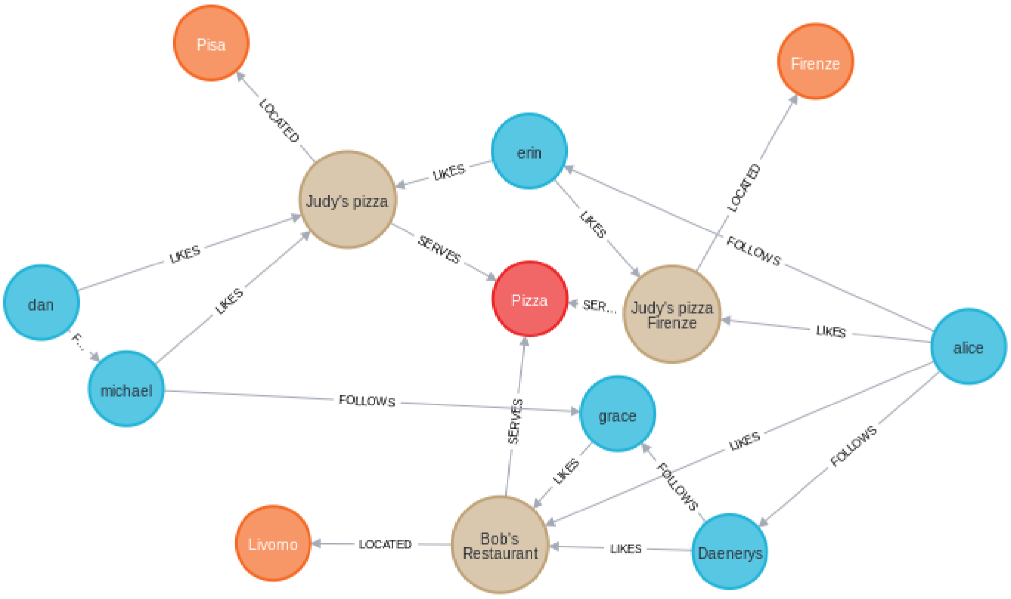


Figure 2.7: Restuarants which serves Pizza.

LIKES relationship we can see that effectively Bob's Restaurant and Judy's Pizza have 3 likes each other, instead Judy's Pizza Florence has only two likes, thus is shown after.

Here we can see the results in json:

```
    [
  {
    "r": {
"identity": 35,
"labels": [
        "Restaurant"
      ],
"properties": {
"name": "Bob's Restaurant",
"description": "Pizza <3",
"price": "MIDDLE"
      }
    },
    "likes": 3
  },
  {
    "r": {
"identity": 48,
"labels": [
        "Restaurant"
      ],
"properties": {
```

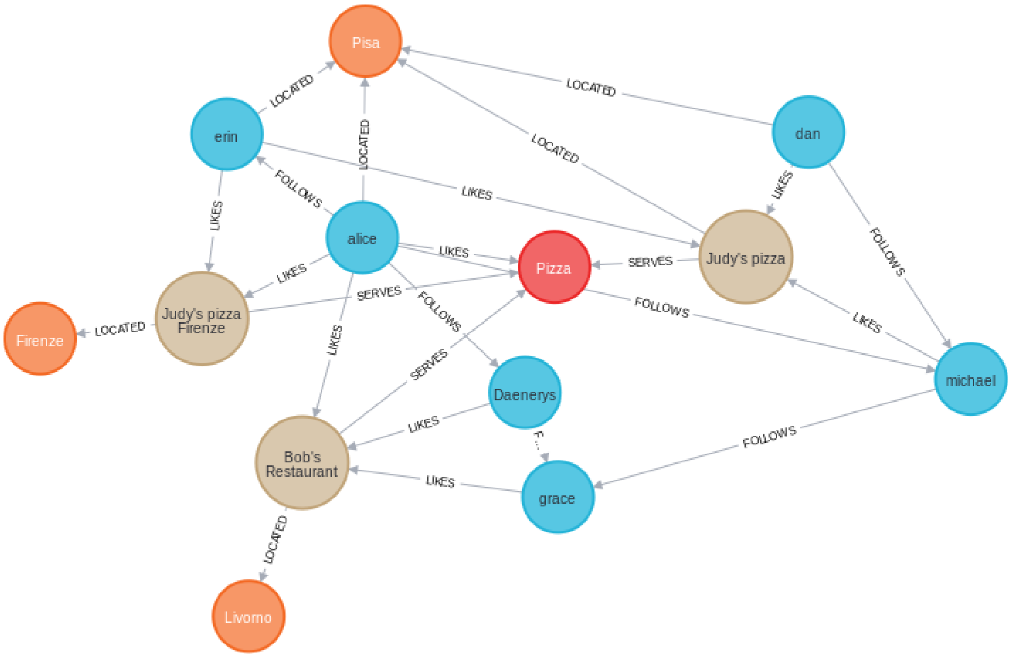


Figure 2.9: Graph snapshot for restaurant recommendations.

```
"name": "Judy’s pizza",
"description": "The best pizzas of the world!",
"price": "HIGH"
      }
    },
    "likes": 3
  },
  {
    "r": {
"identity": 49,
"labels": [
        "Restaurant"
      ],
"properties": {
"name": "Judy’s pizza Firenze",
"description": "The best pizzas of the world!",
"price": "HIGH"
      }
    },
    "likes": 2
  }
]
```

3 | User Manual

3.1 Introduction

Ristogo is an application used to recommend restaurants to people. Every user can be only a customer or also a restaurateur, which can manage more functionalities.

Every user can follow other users as in a social application and can also put like to his/her favourite restaurants.

Information of followers and likes are used to recommend restaurants to the user, using his/her friends preferences.

Restaurateurs can manage their restaurants modifying information in every moment and can see the position (calculated with the number of likes) in the restaurant's ranking with the same cuisine or in the same city.

To run the server:

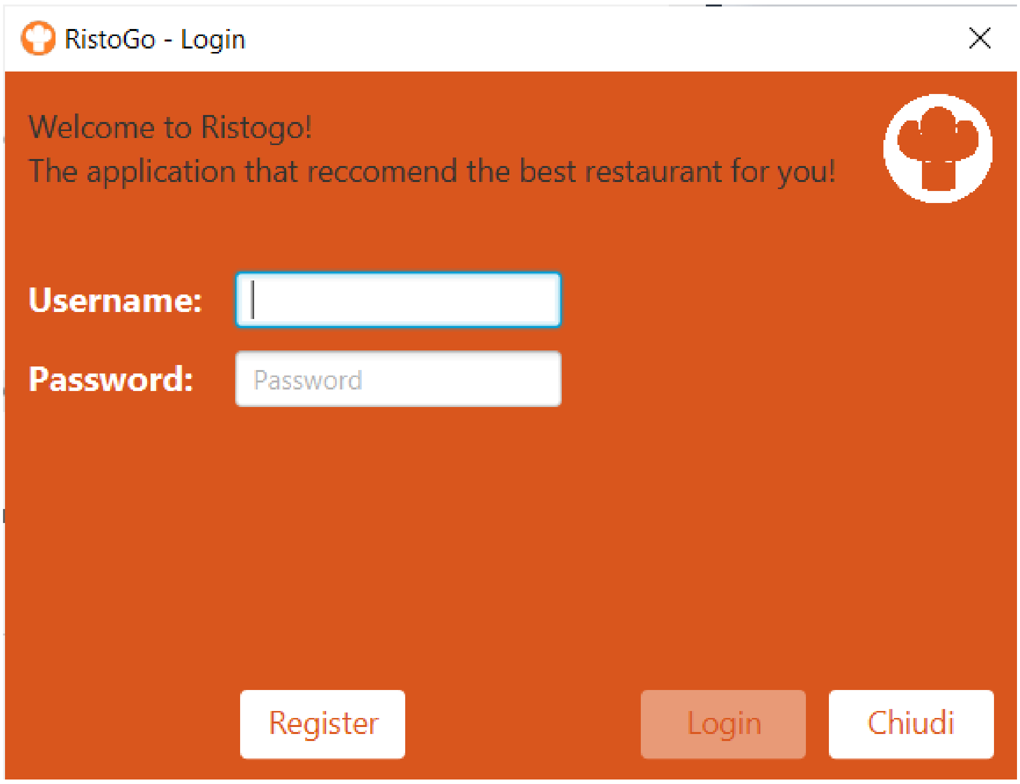
```
$ java -jar RistogoServer.jar
```

To run the client:

```
$ java -jar Ristogo.jar
```

The `-h` option can be used to view all the command line options available. By default, the server expect to find a local Neo4j database with the password `password`.

3.2 Login and Registration



Welcome to Ristogo!
The application that reccomend the best restaurant for you!

Figure 3.1: Login interface.

When the application starts it opens the main page (Figure 3.1), on which an old user can insert username and password to login, clicking on the button "Login".

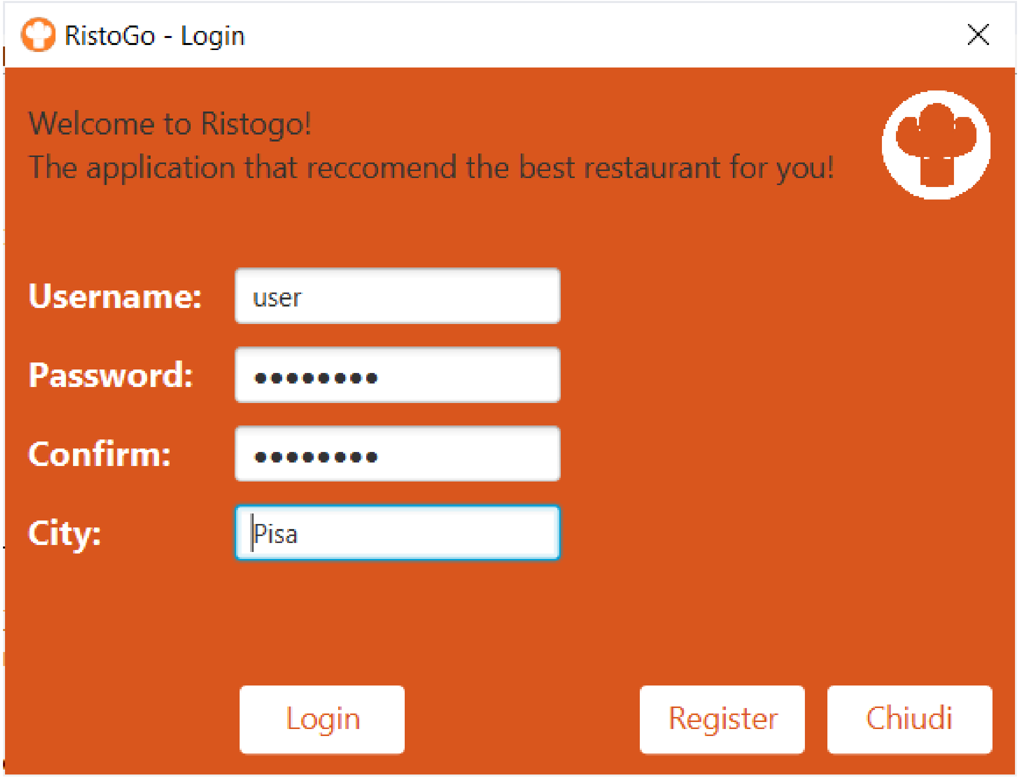
Otherwise, if the user is a new one, he must do the registration procedure to use the application.

If you want to close the application, click on the button "Close".

3.2.1 Registration

If you are a new user, click on "Register" on the main page to open the registration form (Figure 3.2).

Then, insert your username and your password (the password must contain at least 8 characters). Insert another time your password to confirm and then the city where you lives If you went in the registration page wrongly, click on "Login" to return to the login page.



RistoGo - Login

Welcome to Ristogo!
The application that reccomend the best restaurant for you!

Username: user

Password: ●●●●●●●●●

Confirm: ●●●●●●●●●

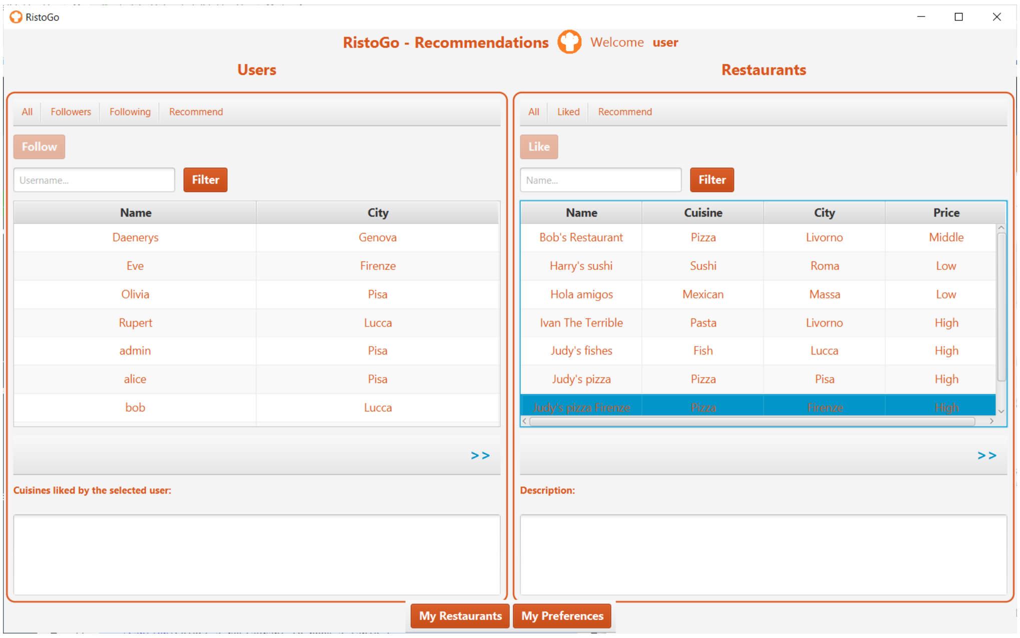
City: Pisa

Figure 3.2: Registration of a user.

When you filled out the form, click on "Register". Now if you don't receive error messages your account will be create. So you can start to use the application for the first time.

3.3 Customer interface

When you are logged in the main page you can see first the customer interface, composed by a section for users and another for restaurants (Figure 3.3).



Name	City
Daenerys	Genova
Eve	Firenze
Olivia	Pisa
Rupert	Lucca
admin	Pisa
alice	Pisa
bob	Lucca

Name	Cuisine	City	Price
Bob's Restaurant	Pizza	Livorno	Middle
Harry's sushi	Sushi	Roma	Low
Hola amigos	Mexican	Massa	Low
Ivan The Terrible	Pasta	Livorno	High
Judy's fishes	Fish	Lucca	High
Judy's pizza	Pizza	Pisa	High
Judy's pizza Firenze	Pizza	Firenze	High

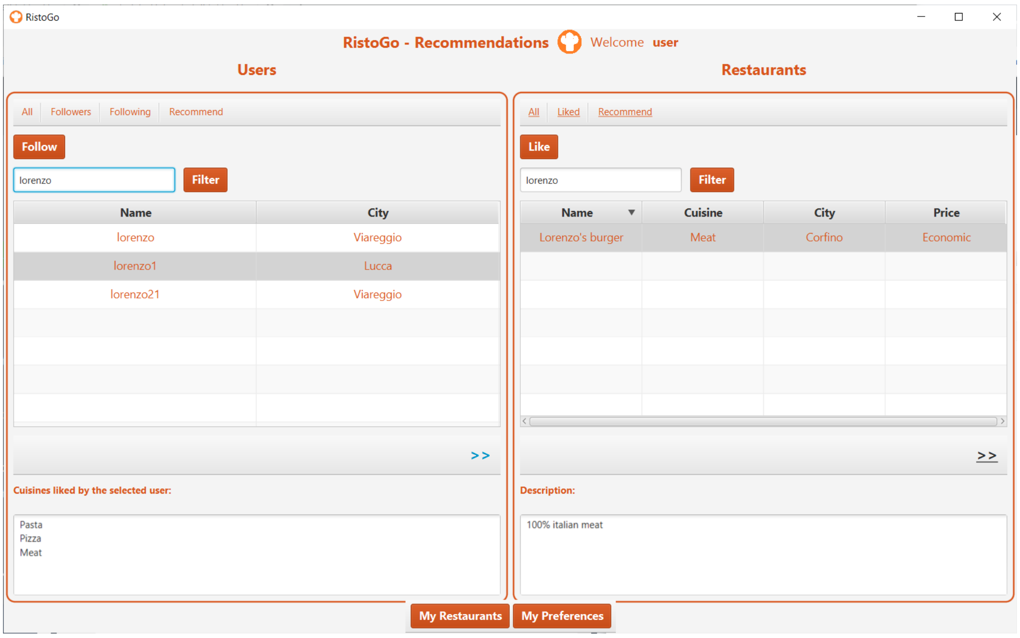
Figure 3.3: Customer interface.

3.3.1 Follow users or put like to restaurants

If you want to search particular users or restaurants, insert the name (or part of the name) in the text field near the corresponding button "Find" and then click it (Figure 3.4). Otherwise, don't do nothing and see the whole list.

You can follow a user by selecting him (you can see also his/her favourite cuisines) and clicking the button "Follow". If there are no errors you can see the user in your following list.

You can put like to a restaurant by selecting it (you can see its description) and clicking on the button "Like". If there are no errors you can see the user in your like list.



lorenzo

Name	City
lorenzo	Viareggio
lorenzo1	Lucca
lorenzo21	Viareggio

lorenzo

Name	Cuisine	City	Price
Lorenzo's burger	Meat	Corfino	Economic

Cuisines liked by the selected user:

Pasta
Pizza
Meat

Description:

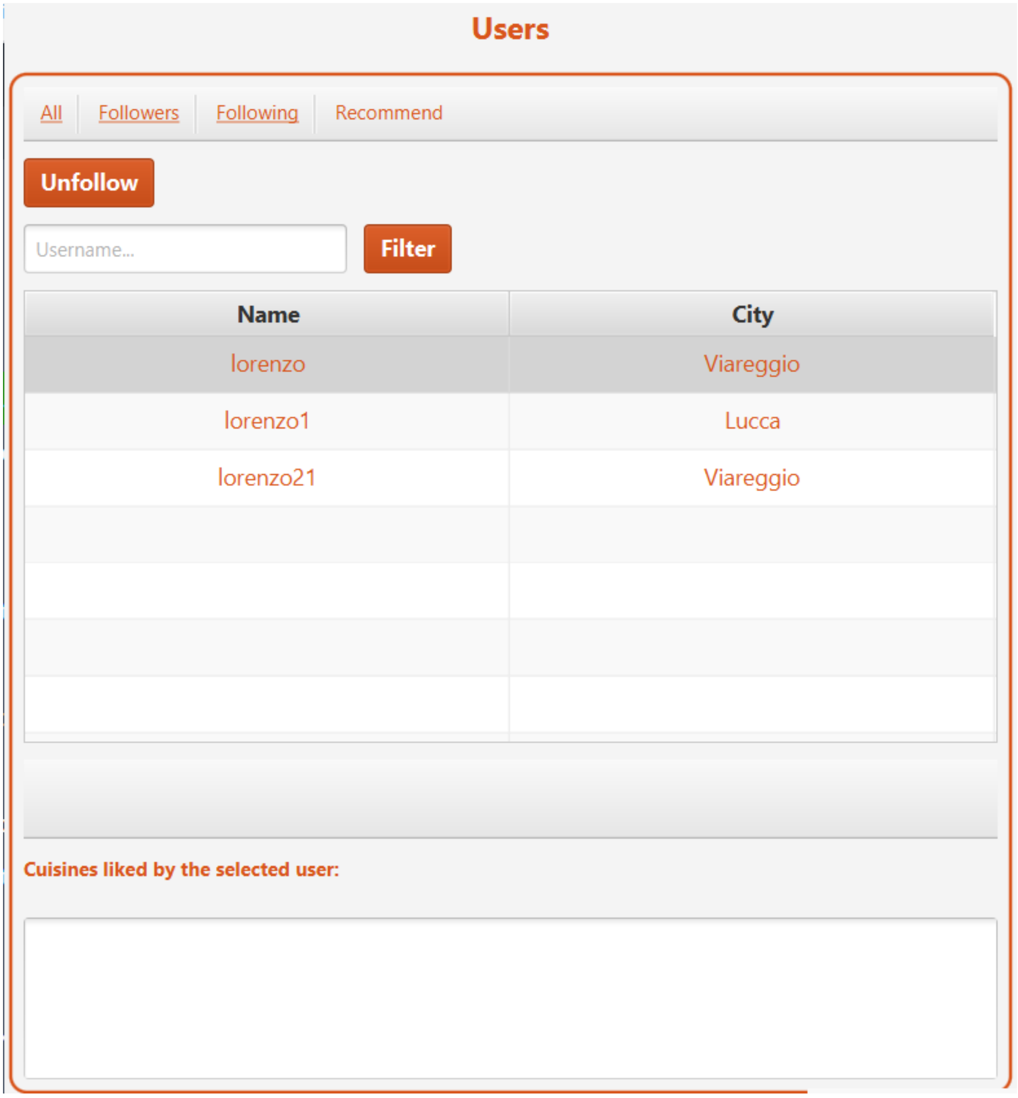
100% italian meat

Figure 3.4: Search users or restaurants.

3.3.2 Unfollow users

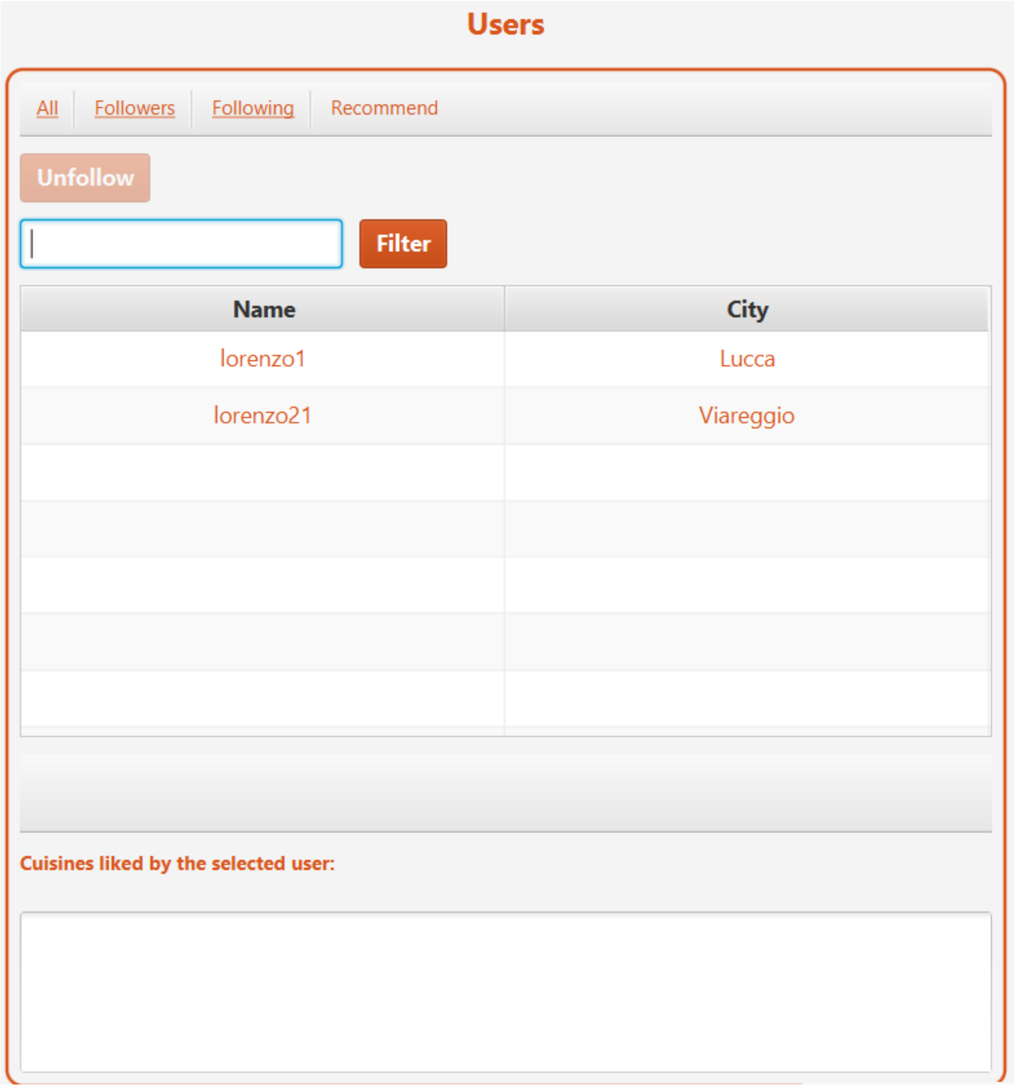
In the user section, open the following list by clicking on “following”. Here you can find all the people that you follow. If you want, you can also see the list of your followers clicking on “followers” (Figure 3.5).

Then, you can select one of your friends and click “unfollow” to stop following him/her. If there are no errors, now you can't see the user in your following list (Figure 3.6).



Name	City
lorenzo	Viareggio
lorenzo1	Lucca
lorenzo21	Viareggio

Figure 3.5: List of users followed.



Name	City
lorenzo1	Lucca
lorenzo21	Viareggio

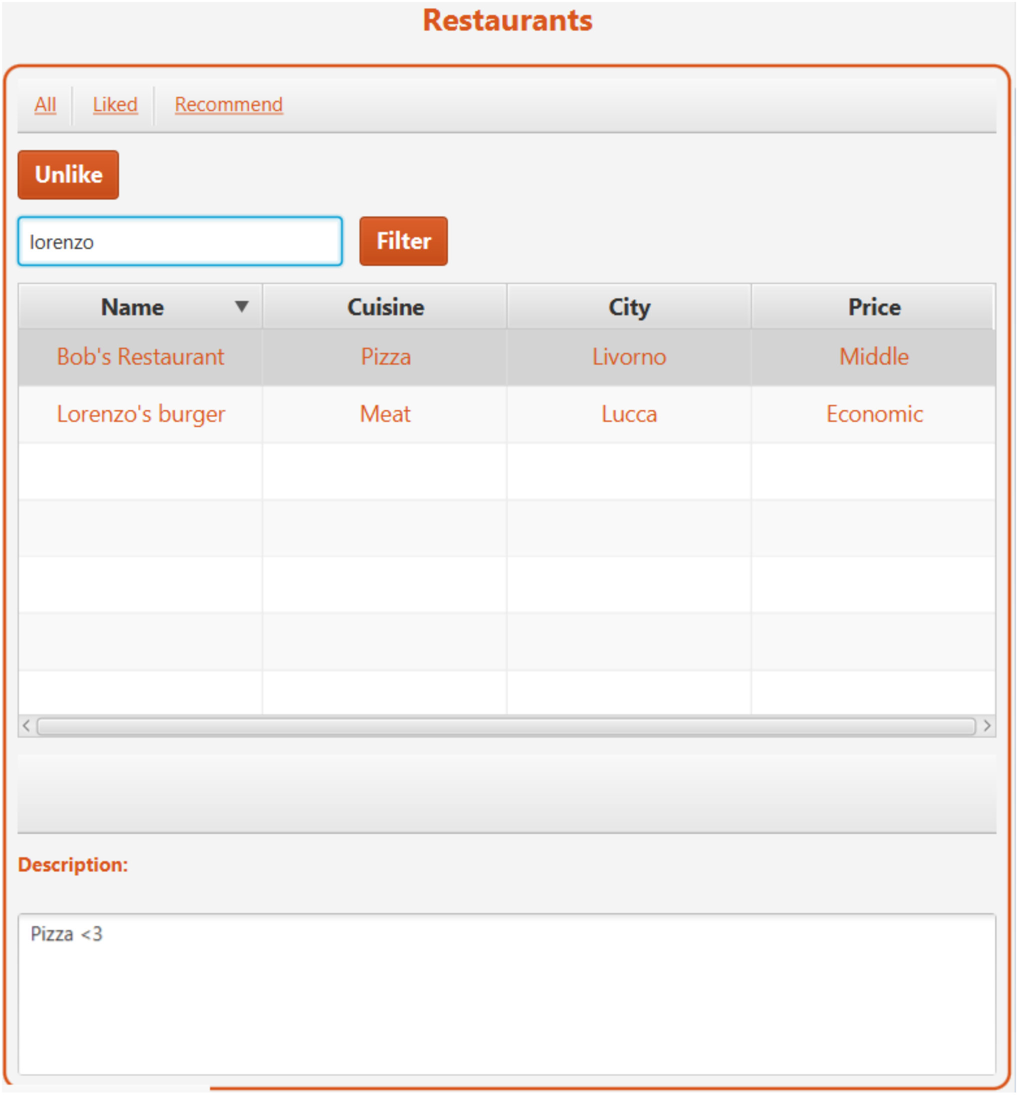
Cuisines liked by the selected user:

Figure 3.6: List of users followed after an unfollow operation.

3.3.3 Unlike restaurants

In the restaurant section, open the like list by clicking on “liked”. Here you can find all the restaurants that you like. (Figure 3.7).

Then, you can select one of your favourite restaurants and click “unlike” to stop like it. If there are no errors, now you can’t see the restaurant in your like list (Figure 3.8).



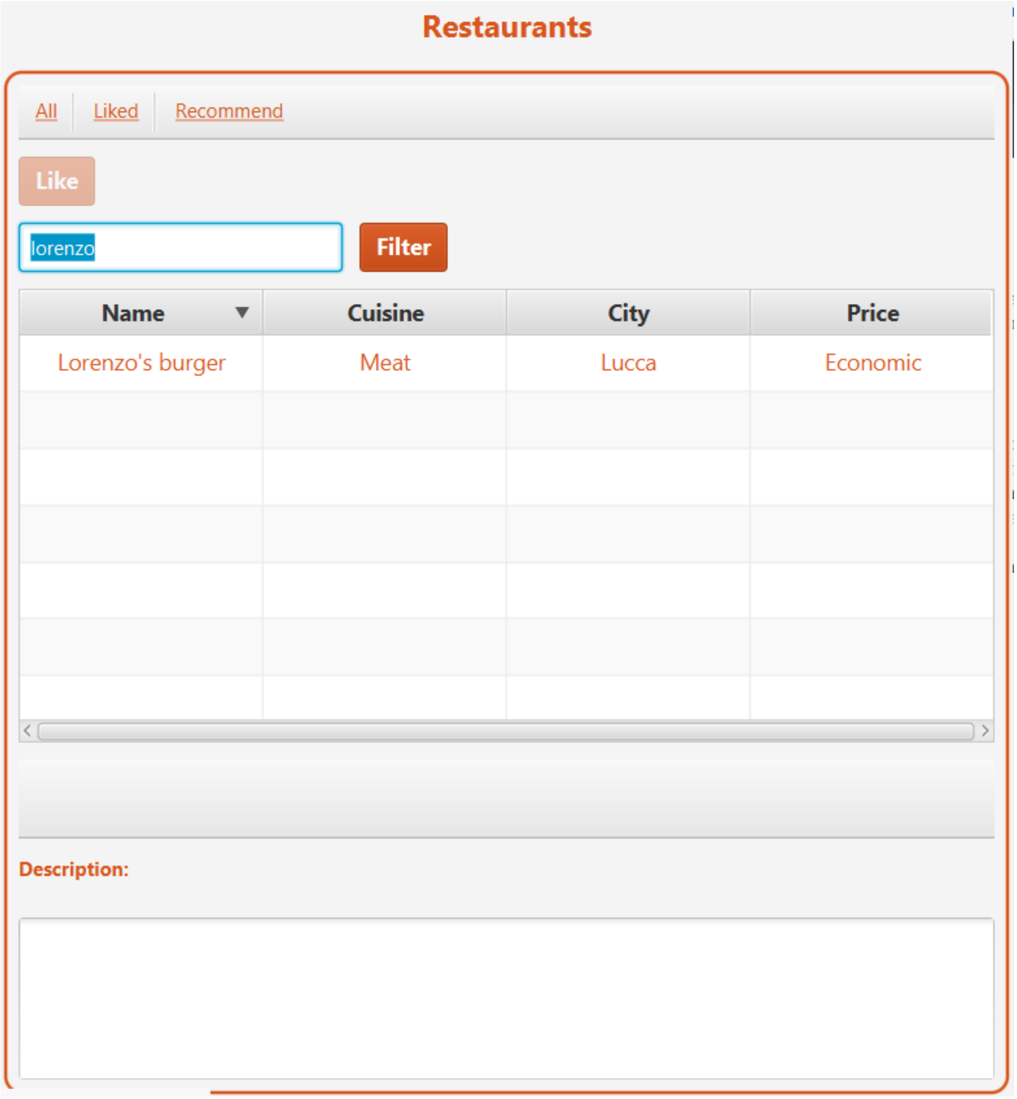
lorenzo

Name	Cuisine	City	Price
Bob's Restaurant	Pizza	Livorno	Middle
Lorenzo's burger	Meat	Lucca	Economic

Description:

Pizza <3

Figure 3.7: List of liked restaurants.



lorenzo

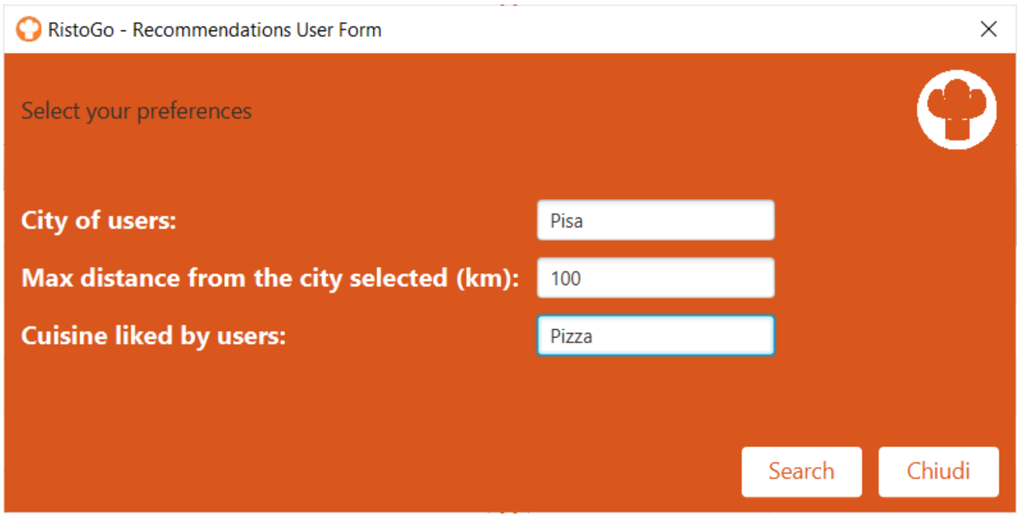
Name	Cuisine	City	Price
Lorenzo's burger	Meat	Lucca	Economic

Figure 3.8: List of liked restaurants after an unlike operation.

3.3.4 Friends recommendations

In the user section, click on “recommend” to open the recommendation page (Figure 3.9).

Then, select a city (by default it is the user city) and the maximum distance from that city (if you want to search into all the cities, write a big value such as 10000). You can optionally select a type of cuisine. So you can filter all the users that you are not already following, which are located into a circle with center city and radius maximum distance, that likes the selected cuisine, if specified (Figure 3.10).



RistoGo - Recommendations User Form

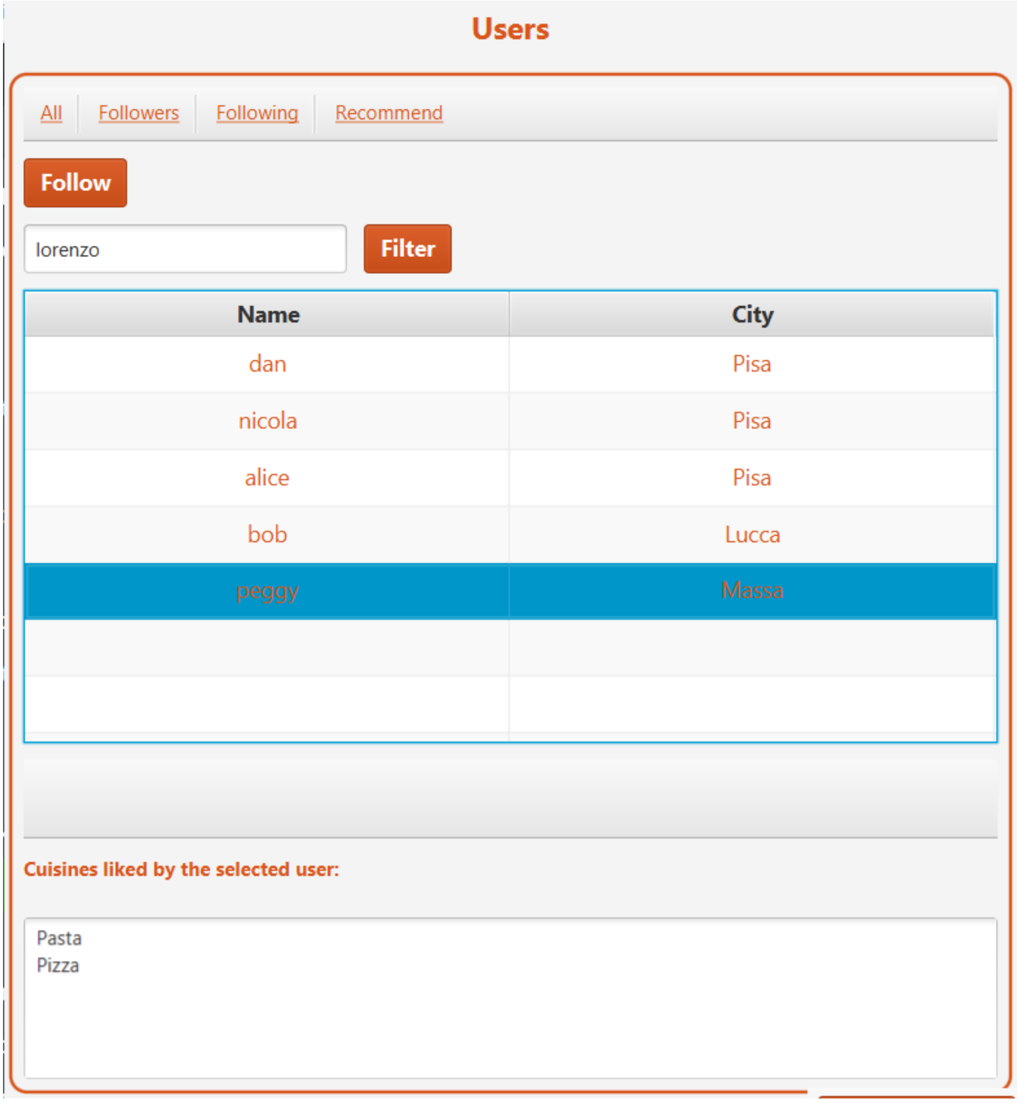
Select your preferences

City of users: Pisa

Max distance from the city selected (km): 100

Cuisine liked by users: Pizza

Figure 3.9: Form for users recommendations.



lorenzo

Name	City
dan	Pisa
nicola	Pisa
alice	Pisa
bob	Lucca
peggy	Massa

Cuisines liked by the selected user:

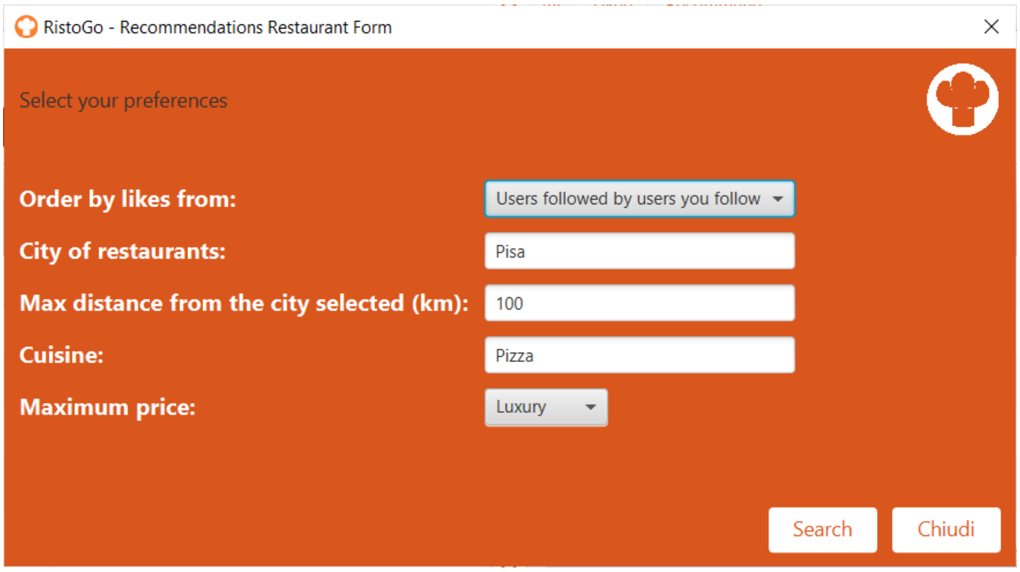
Pasta
Pizza

Figure 3.10: List of users after a recommendation operation.

3.3.5 Restaurants recommendations

In the restaurant section, click on “recommend” to open the recommendation page (Figure 3.11).

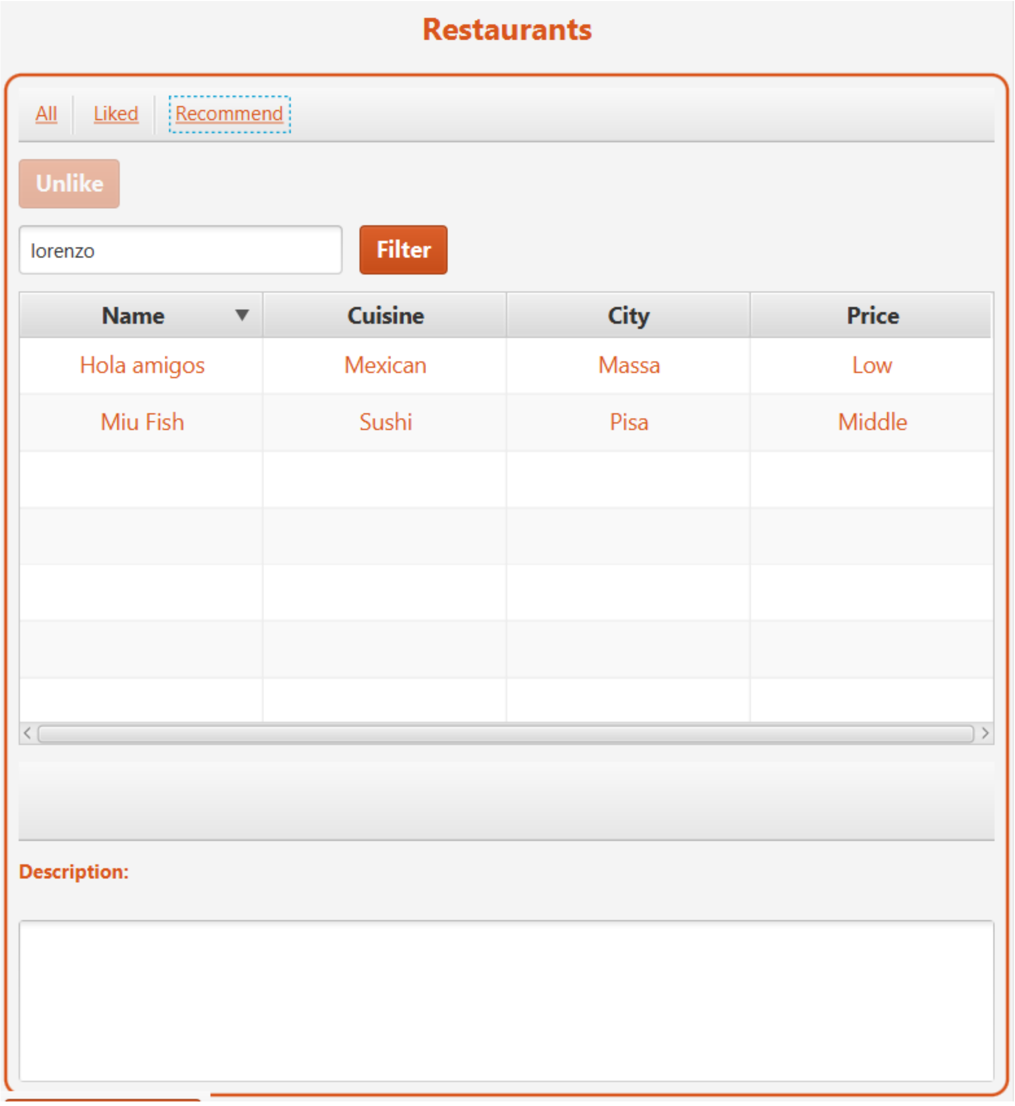
Then, select the order of likes (only friends or also friends of friends), a city (by default it is the user city) and the maximum distance from that city (if you want to search into all the cities, write a big value such as 10000). You can optionally select a type of cuisine. So you can filter all the restaurants liked by the people you selected, that you are not already like, which are located into a circle with center city and radius maximum distance, that present the selected cuisine, if specified (Figure 3.12).


RistoGo - Recommendations Restaurant Form

Select your preferences

Order by likes from:	Users followed by users you follow
City of restaurants:	Pisa
Max distance from the city selected (km):	100
Cuisine:	Pizza
Maximum price:	Luxury

Figure 3.11: Form for restaurants recommendations.



lorenzo

Name	Cuisine	City	Price
Hola amigos	Mexican	Massa	Low
Miu Fish	Sushi	Pisa	Middle

Figure 3.12: List of restaurants after a recommendation operation.

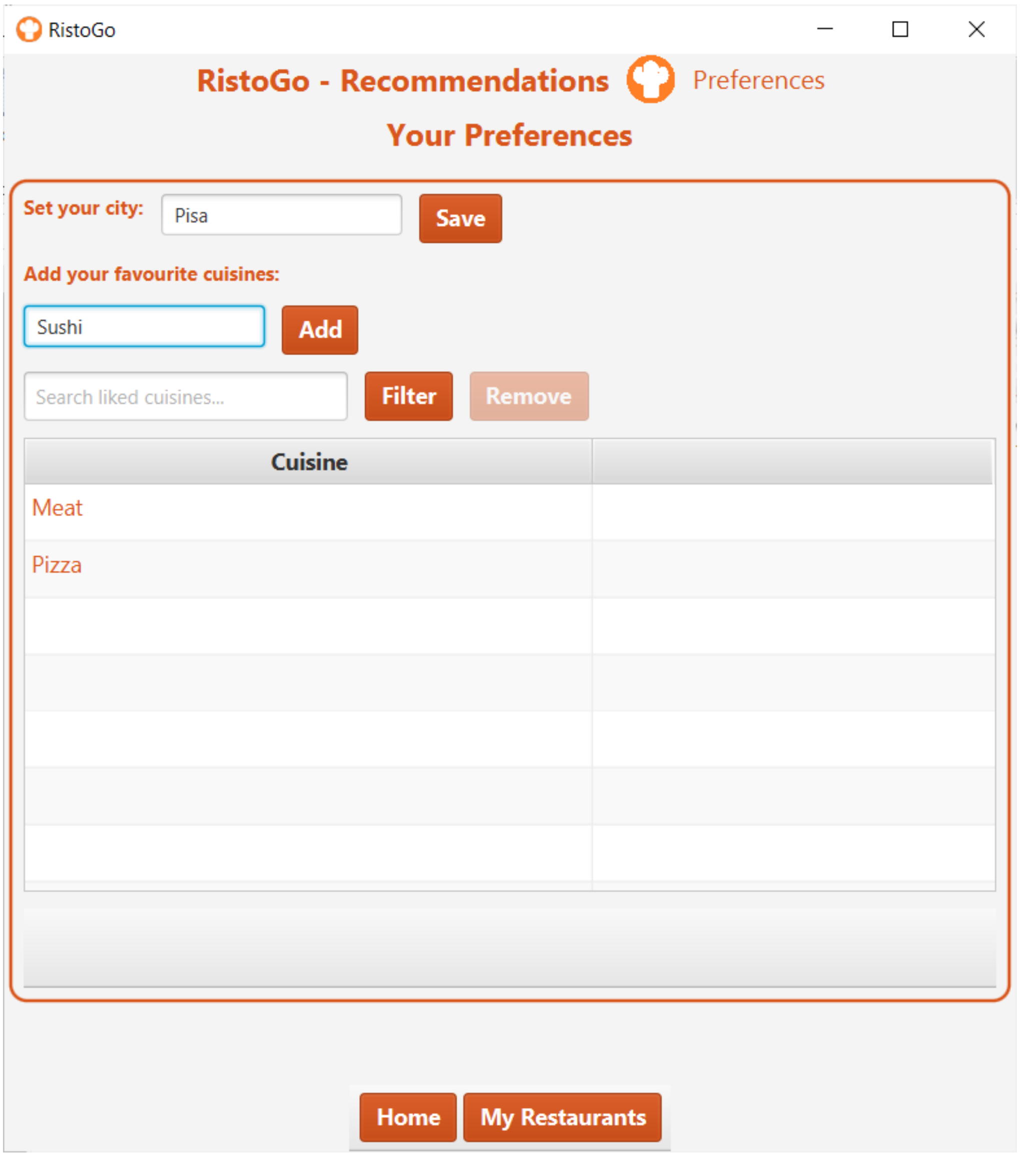
3.3.6 Preferences

In the main page, click on "my preferences" to see your residence city and your favourite cuisines (Figure 3.13).

You can change your residence city typing the new one and clicking "save".

You can also add a new favourite cuisine writing it in the text field and clicking "add". If there are no errors, you can see the new cuisine into the preference list (Figure 3.14).

If you don't like anymore a cuisine you can delete it from your list selecting it and clicking on "remove". If there are no errors, the city now isn't in your preference list (Figure 3.15).

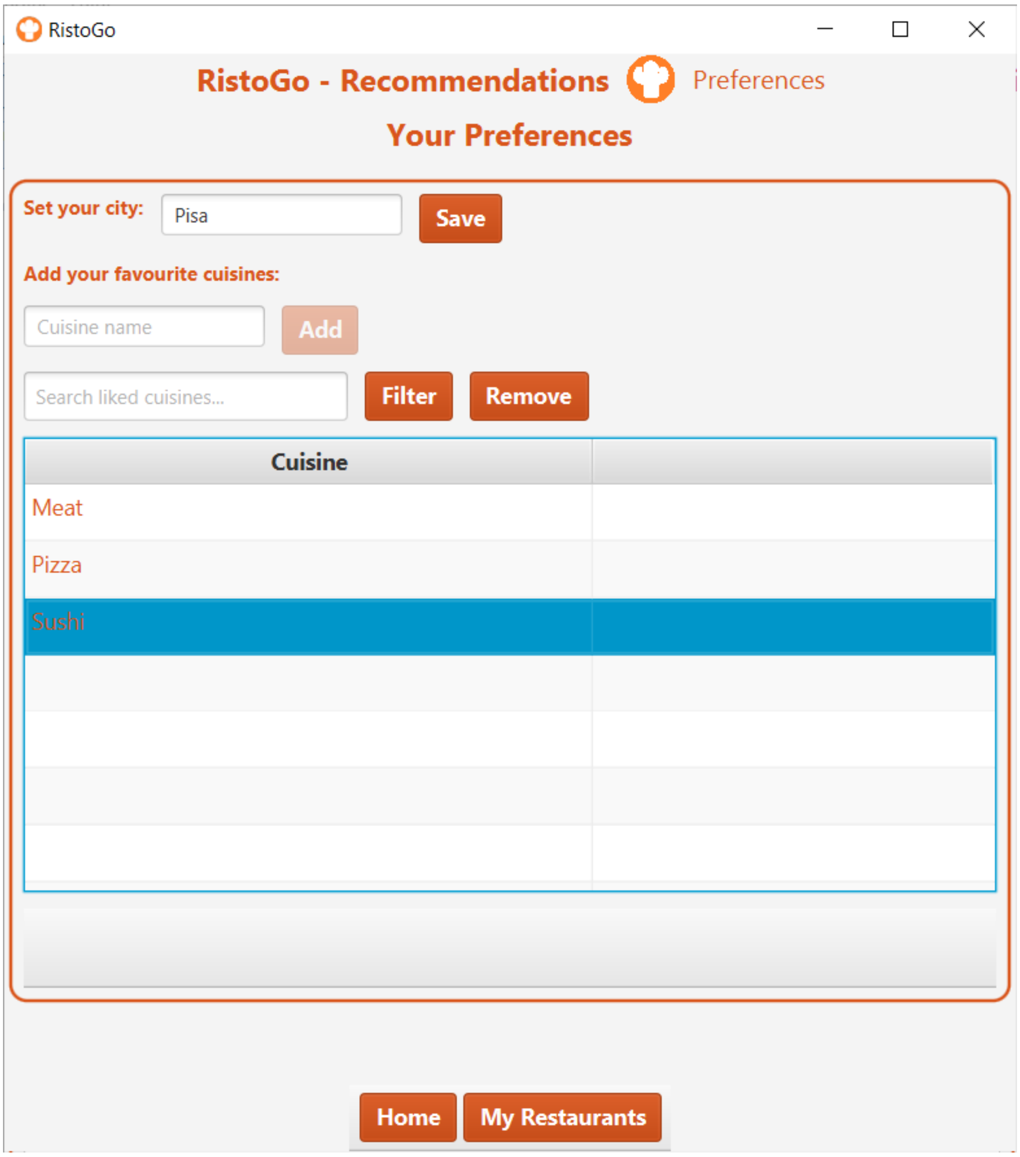


Set your city: Pisa

Add your favourite cuisines: Sushi

Cuisine	
Meat	
Pizza	

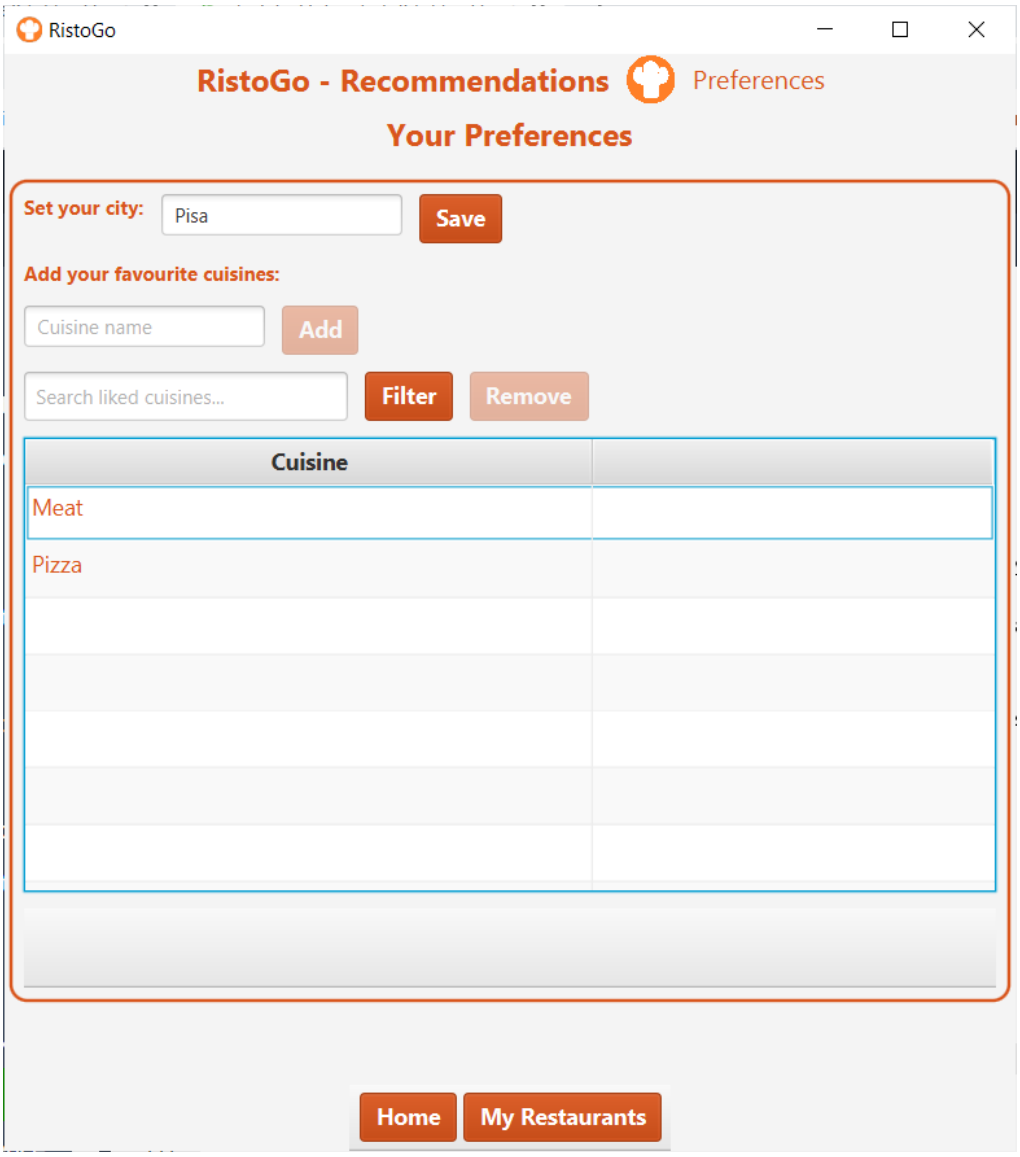
Figure 3.13: Preferences page.



Set your city: Pisa

Cuisine	
Meat	
Pizza	
Sushi	

Figure 3.14: Preferences page after an add operation.



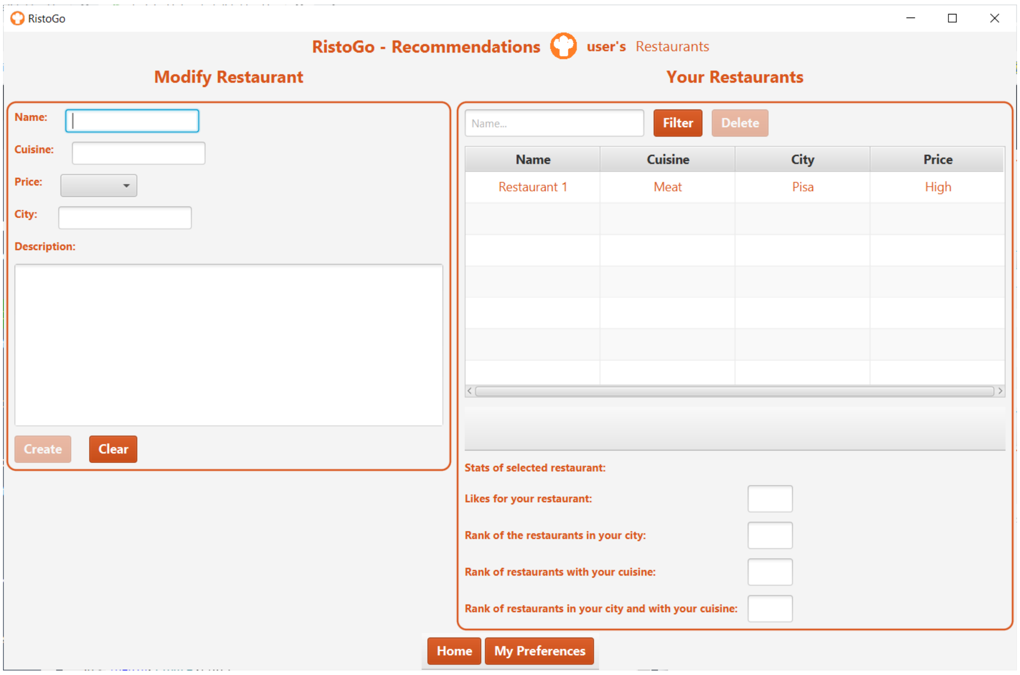
Set your city: Pisa

Cuisine	
Meat	
Pizza	

Figure 3.15: Preferences page after a remove operation.

3.4 Owner interface

To manage your restaurants click on the button “My restaurants” in the main page, this will open a owner interface (Figure 3.16).

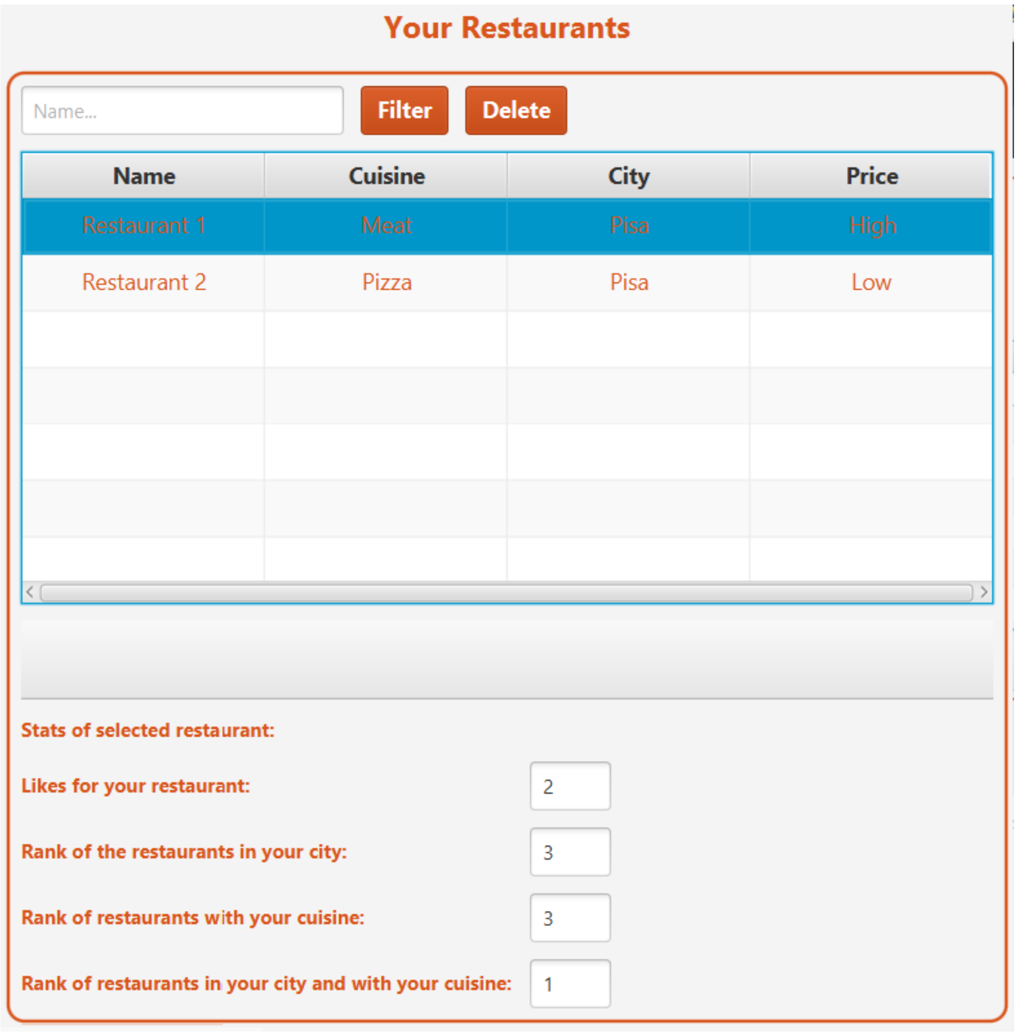

RistoGo
RistoGo - Recommendations
user's Restaurants
Modify Restaurant
Your Restaurants
Name:
Cuisine:
Price:
City:
Description:
Create
Clear
Name...
Filter
Delete
Stats of selected restaurant:
Likes for your restaurant:
Rank of the restaurants in your city:
Rank of restaurants with your cuisine:
Rank of restaurants in your city and with your cuisine:
Home
My Preferences

Name	Cuisine	City	Price
Restaurant 1	Meat	Pisa	High

Figure 3.16: Owner interface.

3.4.1 View restaurant’s statistics

Selecting a restaurant, you can see how many likes your restaurant has and its position considering many rankings: restaurants in the your city, restaurants with your cuisine and restaurants both in your city and with your cuisine (Figure 3.17).



Name	Cuisine	City	Price
Restaurant 1	Meat	Pisa	High
Restaurant 2	Pizza	Pisa	Low

Stats of selected restaurant:

Likes for your restaurant: 2

Rank of the restaurants in your city: 3

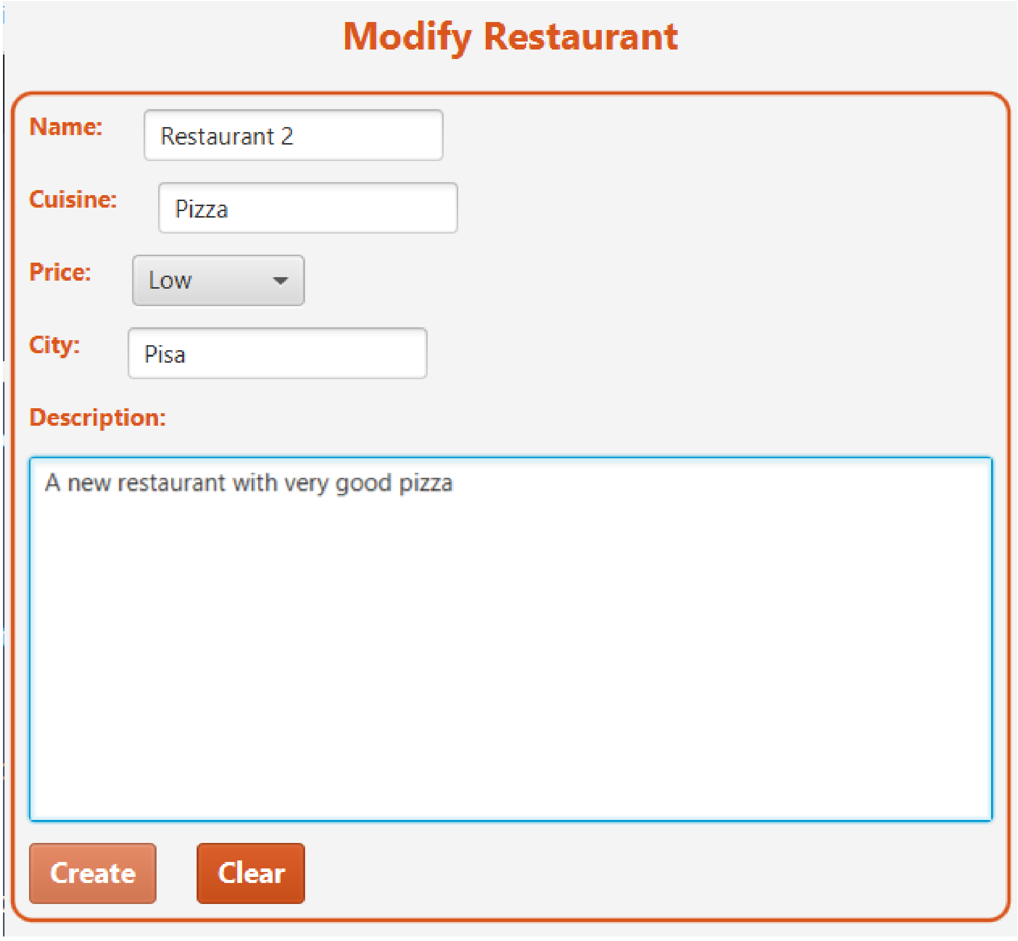
Rank of restaurants with your cuisine: 3

Rank of restaurants in your city and with your cuisine: 1

Figure 3.17: Restaurant's statistics.

3.4.2 Create a new restaurant

To create a new restaurant you have to fill the form of the section "Modify restaurant", writing the name, the cuisine, the city and selecting the price range (Figure 3.18). Then, click on the button "create". If there are no errors you can see the new restaurant into your restaurants list (Figure 3.19).



Modify Restaurant

Name: Restaurant 2

Cuisine: Pizza

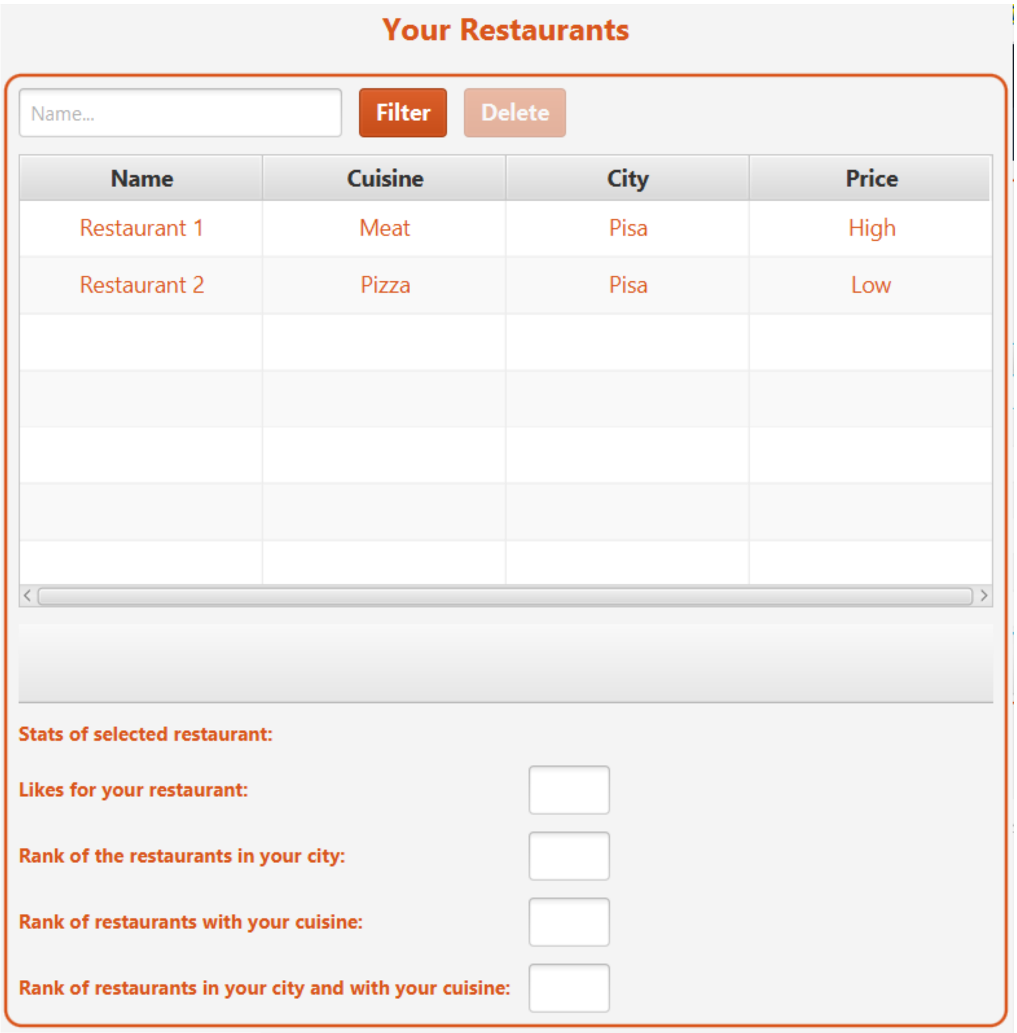
Price: Low

City: Pisa

Description:

A new restaurant with very good pizza

Figure 3.18: Creation of a new restaurant.



Name	Cuisine	City	Price
Restaurant 1	Meat	Pisa	High
Restaurant 2	Pizza	Pisa	Low

Stats of selected restaurant:

Likes for your restaurant:

Rank of the restaurants in your city:

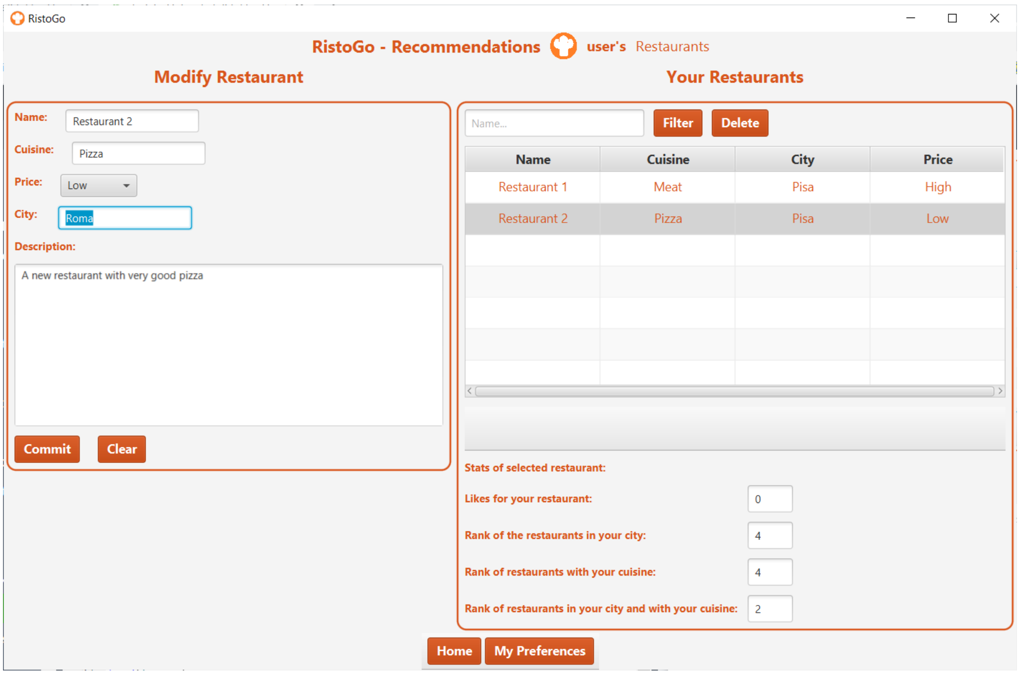
Rank of restaurants with your cuisine:

Rank of restaurants in your city and with your cuisine:

Figure 3.19: List of restaurants after creating a new one.

3.4.3 Modify restaurant’s information

After the first creation you can change your restaurant’s information, selecting it on the Restaurants section and filling the form on the section “Modify your restaurant”. Then click on the button “Commit” (Figure 3.20).



Modify Restaurant

Name: Restaurant 2

Cuisine: Pizza

Price: Low

City: Roma

Description:

A new restaurant with very good pizza

Your Restaurants

Name...

Name	Cuisine	City	Price
Restaurant 1	Meat	Pisa	High
Restaurant 2	Pizza	Pisa	Low

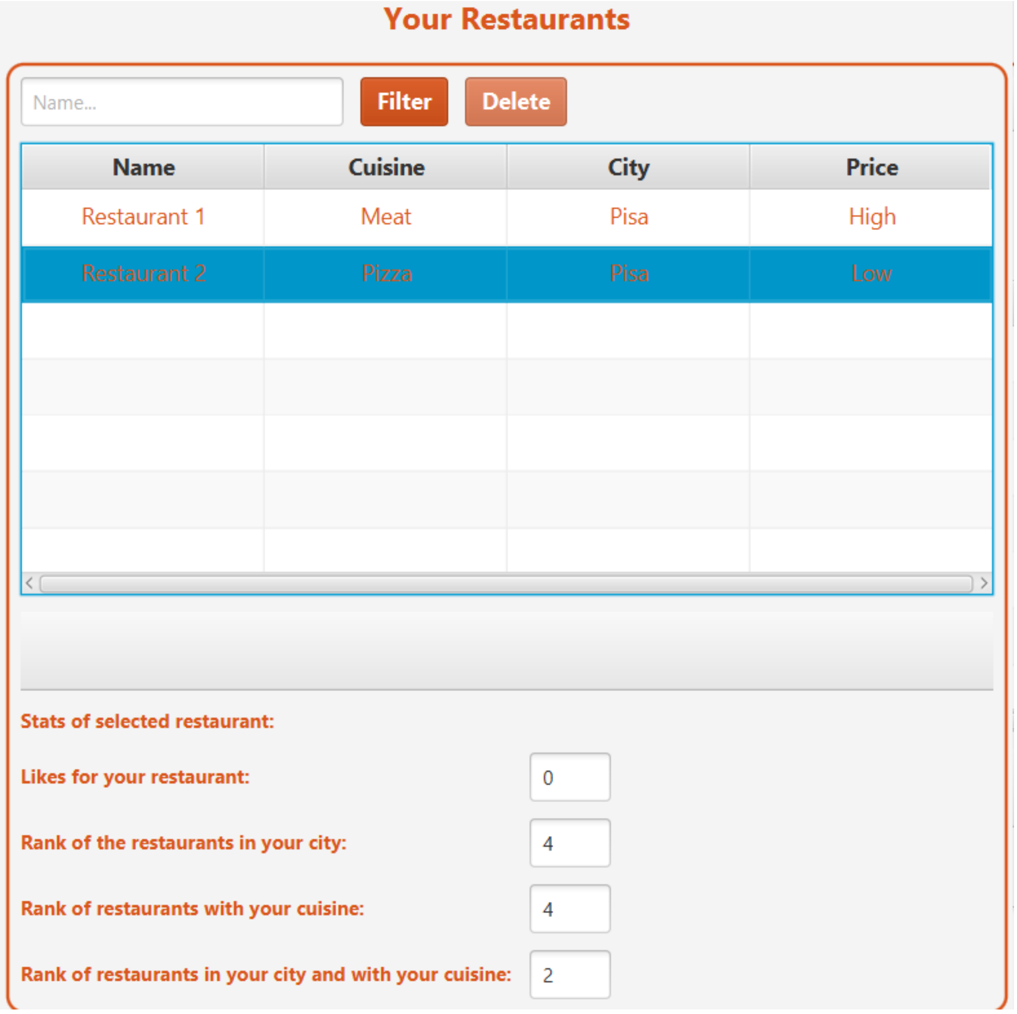
Stats of selected restaurant:

Likes for your restaurant:	0
Rank of the restaurants in your city:	4
Rank of restaurants with your cuisine:	4
Rank of restaurants in your city and with your cuisine:	2

Figure 3.20: Modify restaurant's information.

3.4.4 Delete a restaurant

If a restaurant doesn't exist anymore, you can delete it. Click on the right restaurant and press "delete" (Figure 3.21). If there are no errors you don't see now the restaurant into your list (Figure 3.22)



Your Restaurants

Name	Cuisine	City	Price
Restaurant 1	Meat	Pisa	High
Restaurant 2	Pizza	Pisa	Low

Stats of selected restaurant:

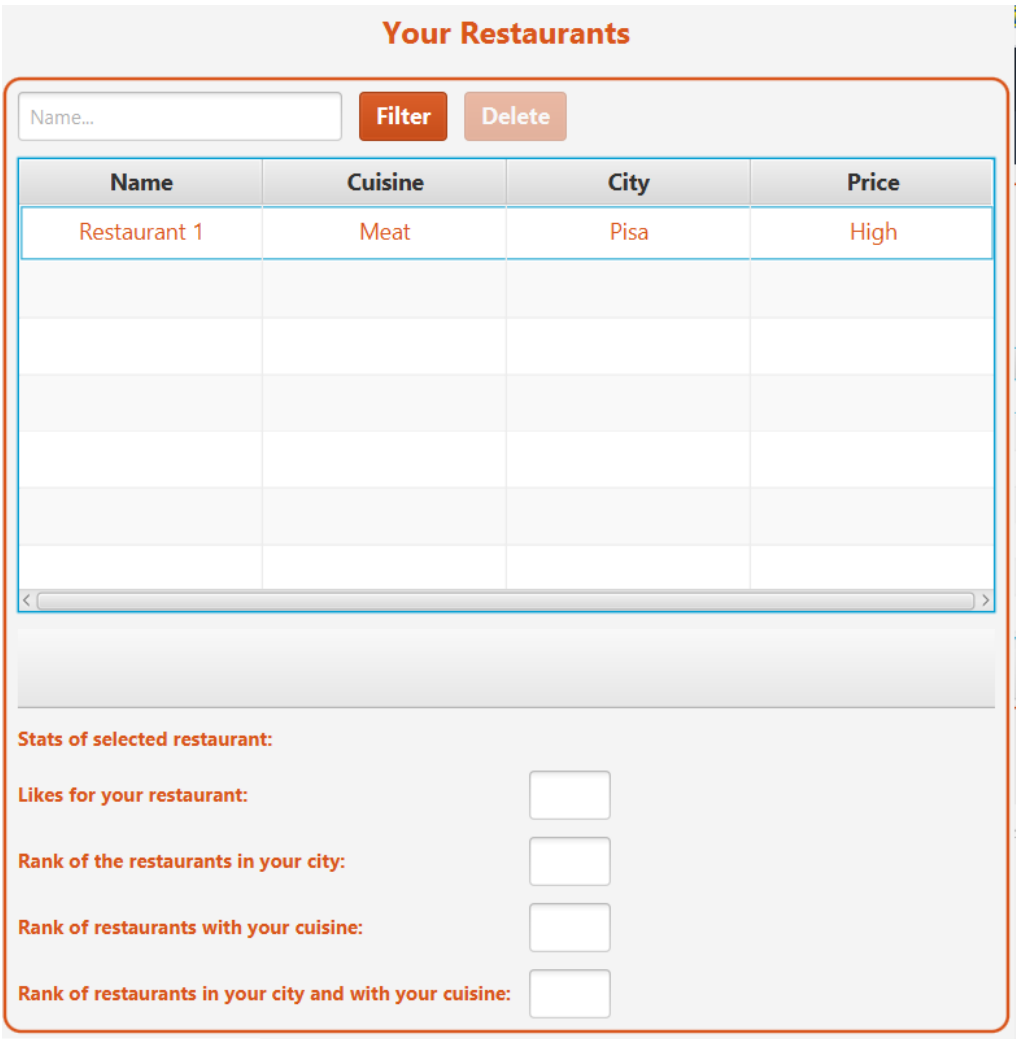
Likes for your restaurant: 0

Rank of the restaurants in your city: 4

Rank of restaurants with your cuisine: 4

Rank of restaurants in your city and with your cuisine: 2

Figure 3.21: Delete a restaurant.



Your Restaurants

Name	Cuisine	City	Price
Restaurant 1	Meat	Pisa	High

Stats of selected restaurant:

Likes for your restaurant:

Rank of the restaurants in your city:

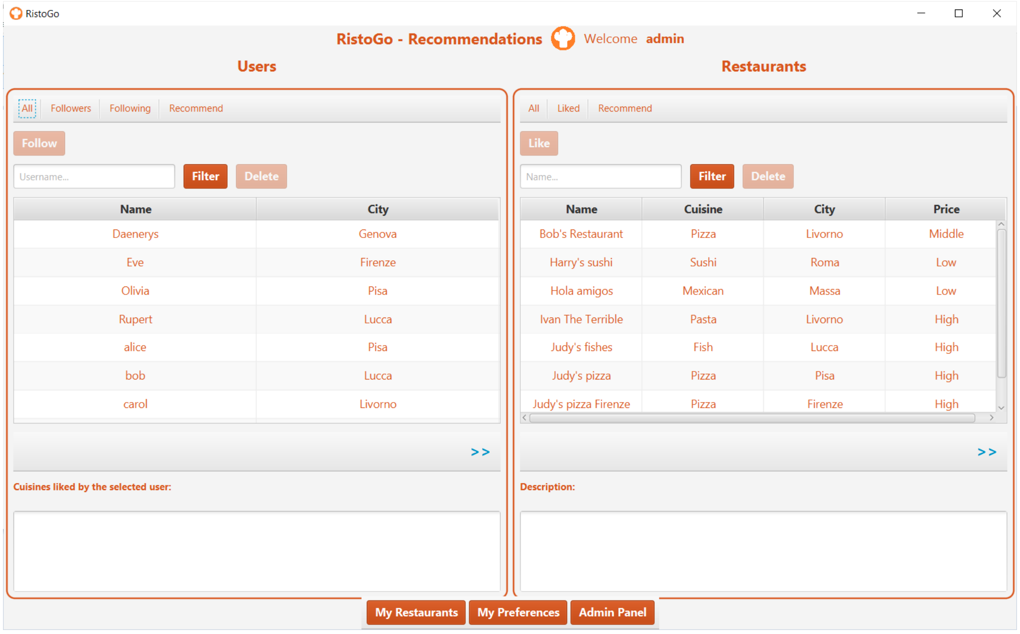
Rank of restaurants with your cuisine:

Rank of restaurants in your city and with your cuisine:

Figure 3.22: List of restaurants after a delete operation.

3.5 Admin interface

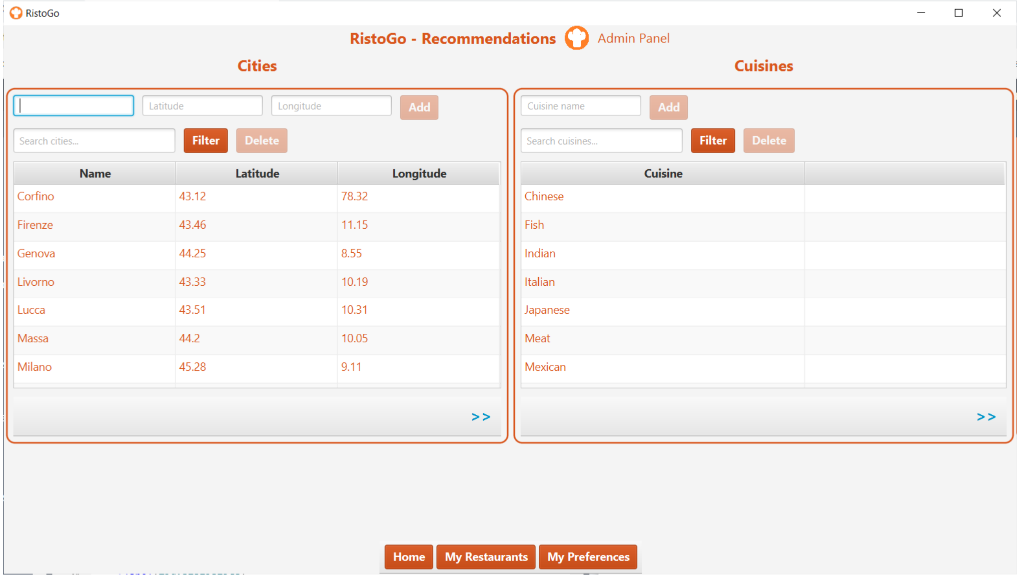
The admin user has all the functionalities of all the normal users and also admin privileges. In the main page he has additional options “delete” and “admin panel” (Figure 3.23). Clicking on “admin panel” you can see the admin page (Figure 3.24).



Name	City
Daenerys	Genova
Eve	Firenze
Olivia	Pisa
Rupert	Lucca
alice	Pisa
bob	Lucca
carol	Livorno

Name	Cuisine	City	Price
Bob's Restaurant	Pizza	Livorno	Middle
Harry's sushi	Sushi	Roma	Low
Hola amigos	Mexican	Massa	Low
Ivan The Terrible	Pasta	Livorno	High
Judy's fishes	Fish	Lucca	High
Judy's pizza	Pizza	Pisa	High
Judy's pizza Firenze	Pizza	Firenze	High

Figure 3.23: Admin main page.



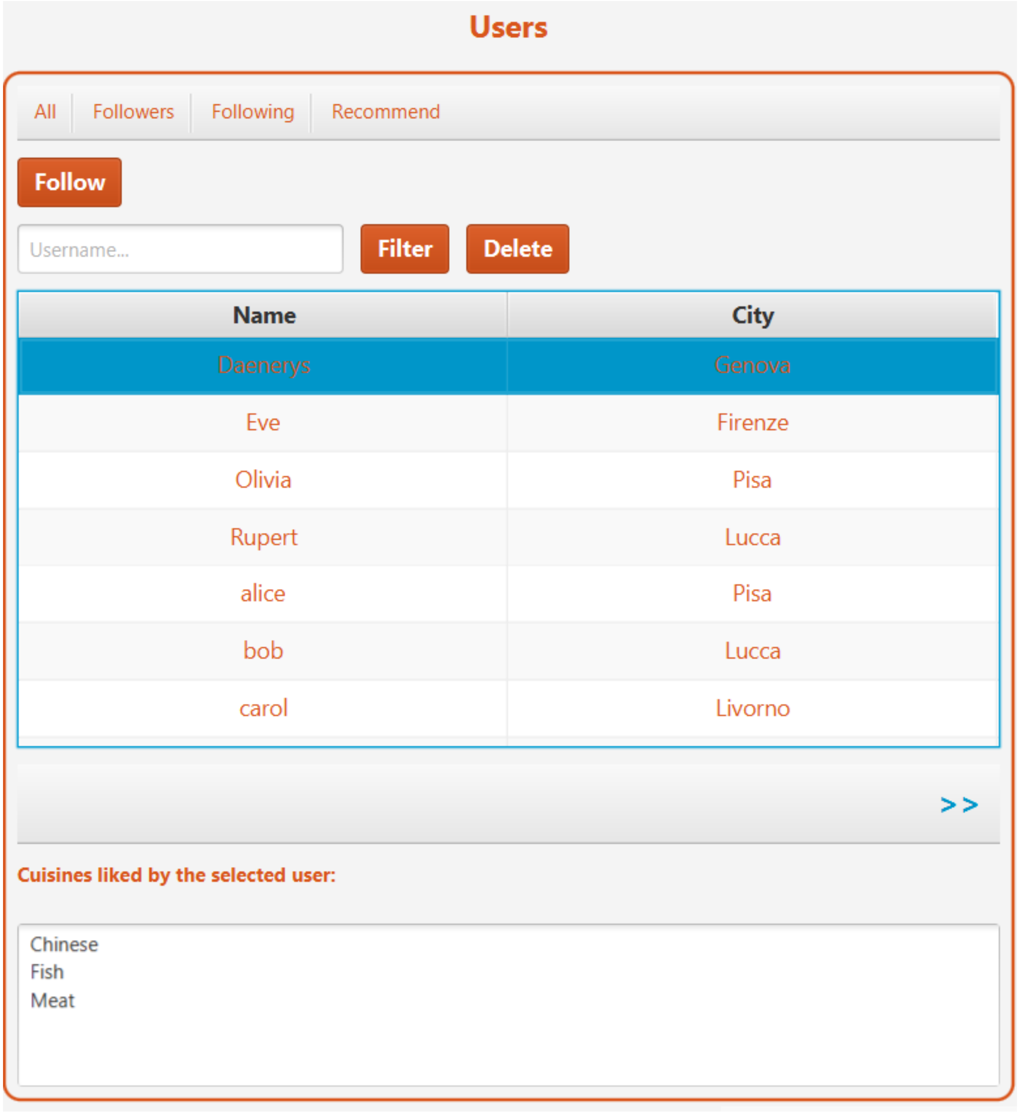
Name	Latitude	Longitude
Corfino	43.12	78.32
Firenze	43.46	11.15
Genova	44.25	8.55
Livorno	43.33	10.19
Lucca	43.51	10.31
Massa	44.2	10.05
Milano	45.28	9.11

Cuisine
Chinese
Fish
Indian
Italian
Japanese
Meat
Mexican

Figure 3.24: Admin panel.

3.5.1 Delete users

From the list, admin can delete a user selecting it from the list and clicking on the button “delete” (Figure 3.25). If there are no errors on the server, no one can now find that user or his restaurants on the lists.



Name	City
Daenerys	Genova
Eve	Firenze
Olivia	Pisa
Rupert	Lucca
alice	Pisa
bob	Lucca
carol	Livorno

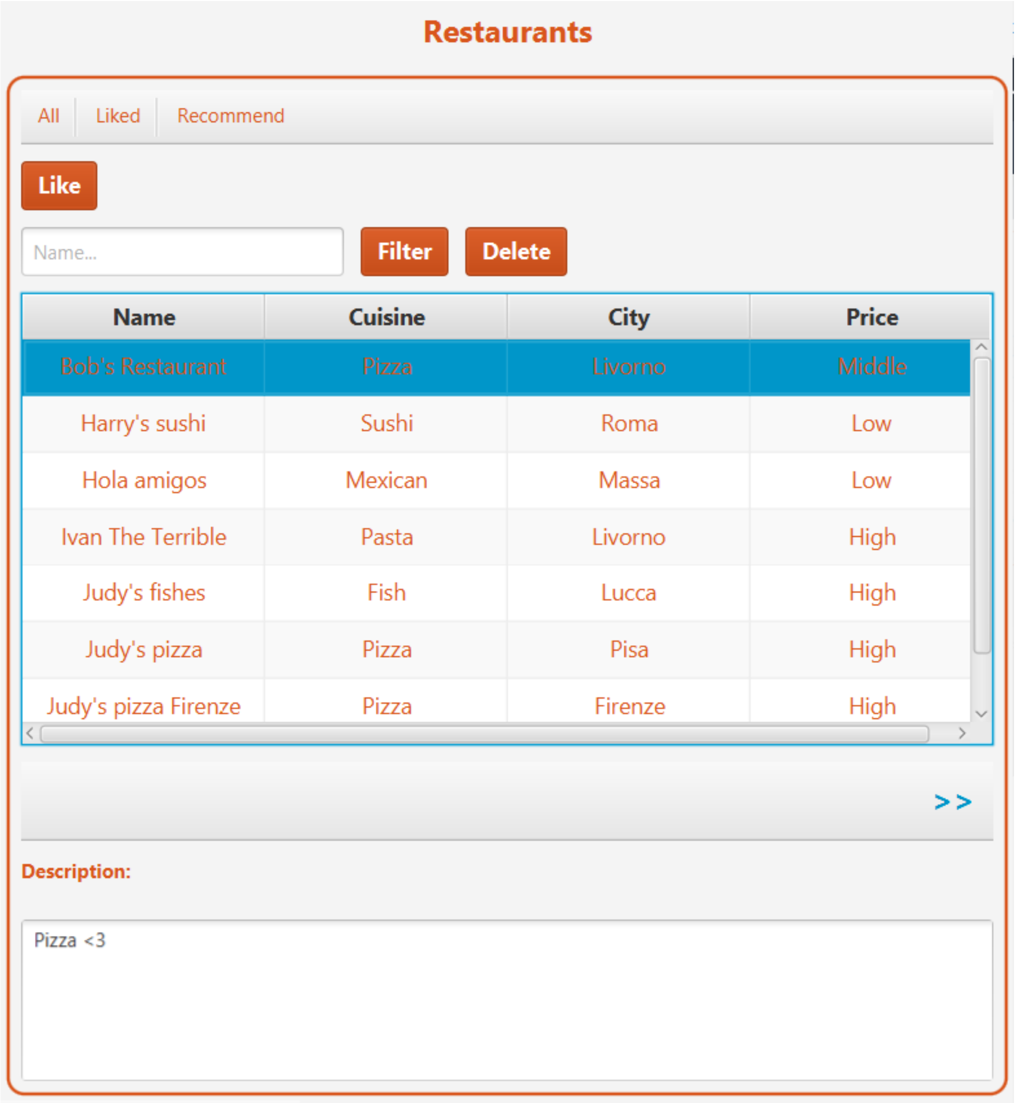
Cuisines liked by the selected user:

Chinese
Fish
Meat

Figure 3.25: Deleting users.

3.5.2 Delete restaurants

From the list, admin can delete a user selecting it from the list and clicking on the button "delete" (Figure 3.26). If there are no errors on the server, no one can now find that restaurant on the lists.



Name	Cuisine	City	Price
Bob's Restaurant	Pizza	Livorno	Middle
Harry's sushi	Sushi	Roma	Low
Hola amigos	Mexican	Massa	Low
Ivan The Terrible	Pasta	Livorno	High
Judy's fishes	Fish	Lucca	High
Judy's pizza	Pizza	Pisa	High
Judy's pizza Firenze	Pizza	Firenze	High

Description:

Pizza <3

Figure 3.26: ADeleting restaurants.

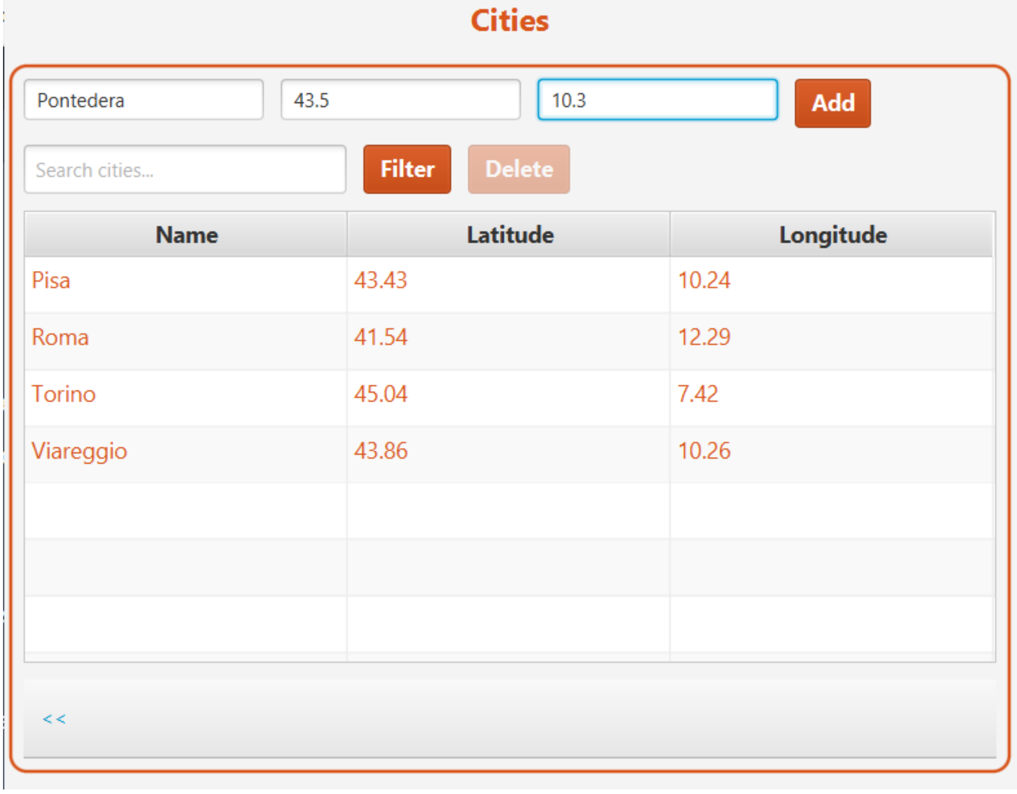
3.5.3 Manage cities

Admin can manage application cities from the admin panel.

He can create a new one inserting its information on the form and then clicking on “add” (Figure 3.27). If there are no errors you can see the city on the list (Figure 3.28).

He can modify a city selecting it, modifying the parameters and clicking on “save” (Figure 3.28). If there are no errors you can see the modified city in the list (Figure 3.29).

He can also delete a city selecting it and clicking on the button “delete” (Figure 3.29). If there are no errors, now the city isn’t anymore on the list and all the users/restaurants located in that city have the city parameter set to “N/D”. (Figure 3.30).

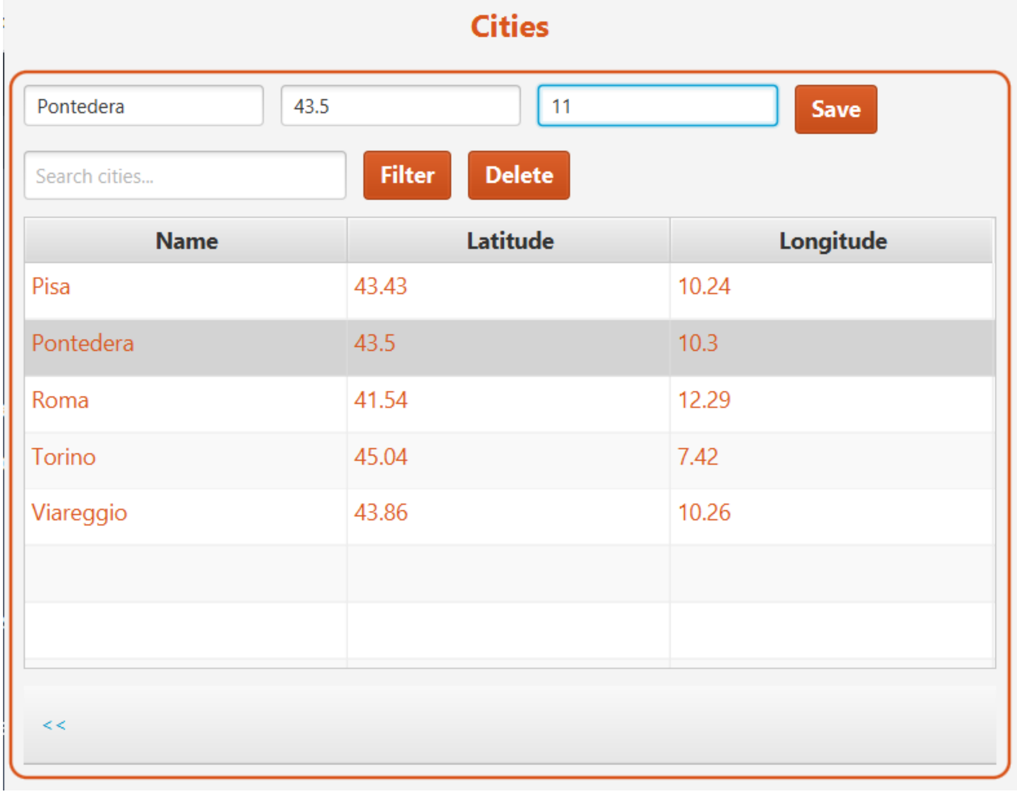


Cities

Pontedera | 43.5 | 10.3

Name	Latitude	Longitude
Pisa	43.43	10.24
Roma	41.54	12.29
Torino	45.04	7.42
Viareggio	43.86	10.26

Figure 3.27: Add a new city.

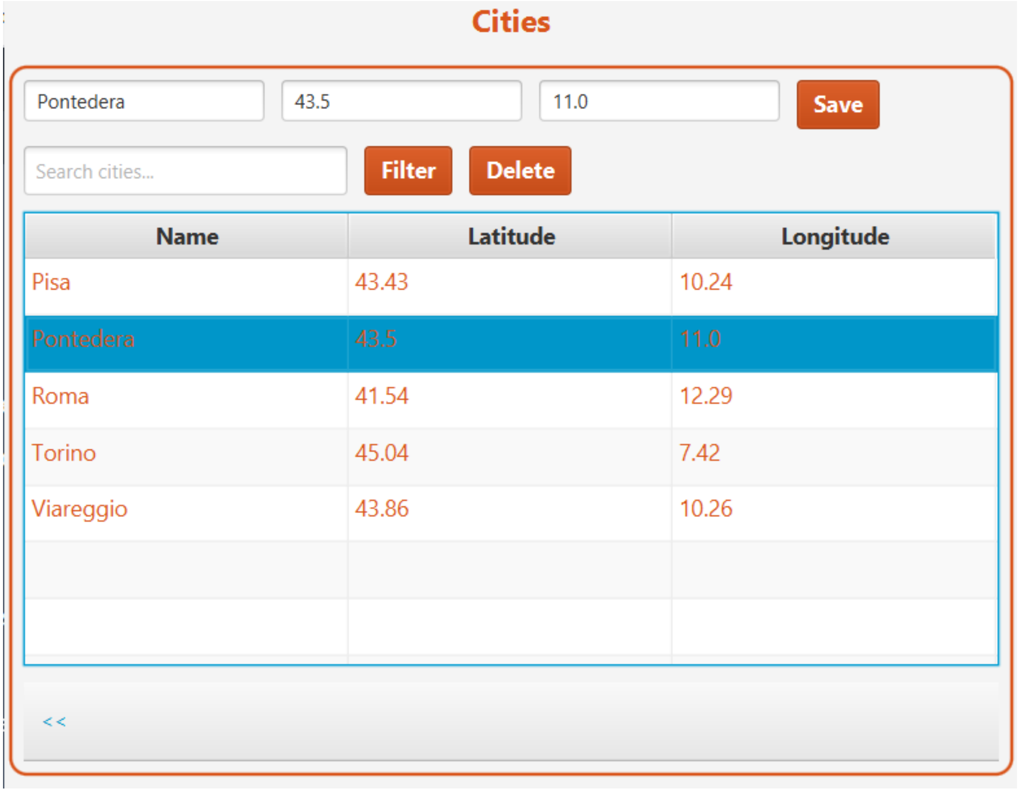


Cities

Pontedera | 43.5 | 11

Name	Latitude	Longitude
Pisa	43.43	10.24
Pontedera	43.5	10.3
Roma	41.54	12.29
Torino	45.04	7.42
Viareggio	43.86	10.26

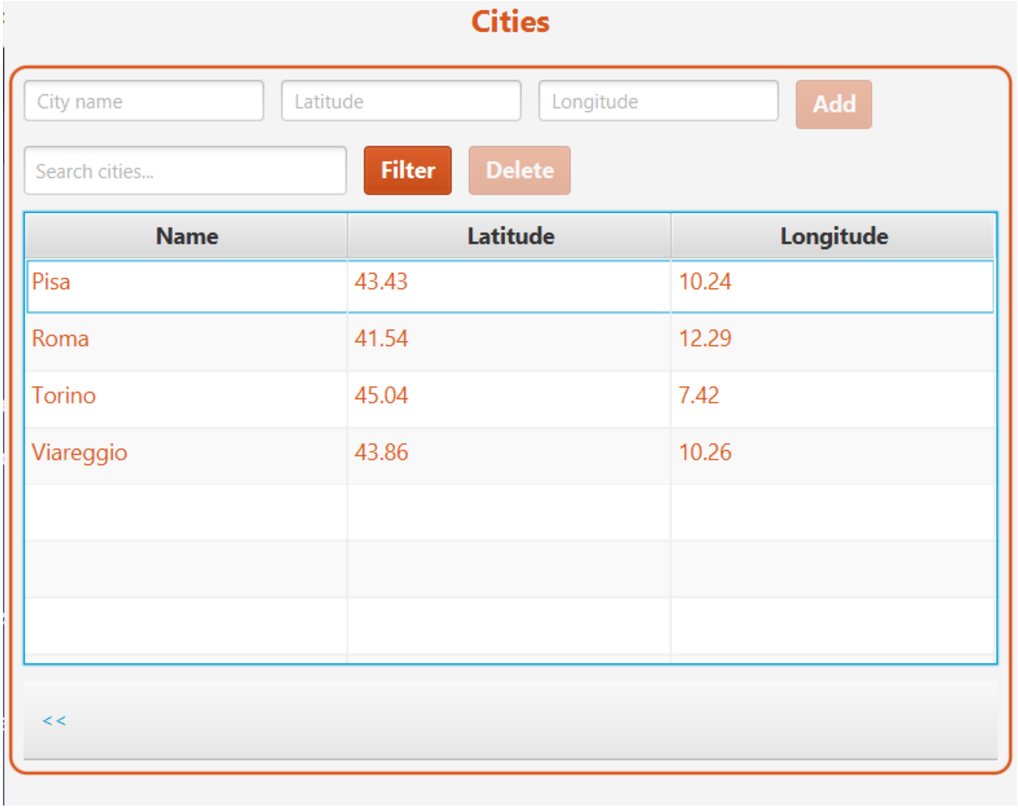
Figure 3.28: Modify a city.



Pontedera | 43.5 | 11.0

Name	Latitude	Longitude
Pisa	43.43	10.24
Pontedera	43.5	11.0
Roma	41.54	12.29
Torino	45.04	7.42
Viareggio	43.86	10.26

Figure 3.29: Delete a city.



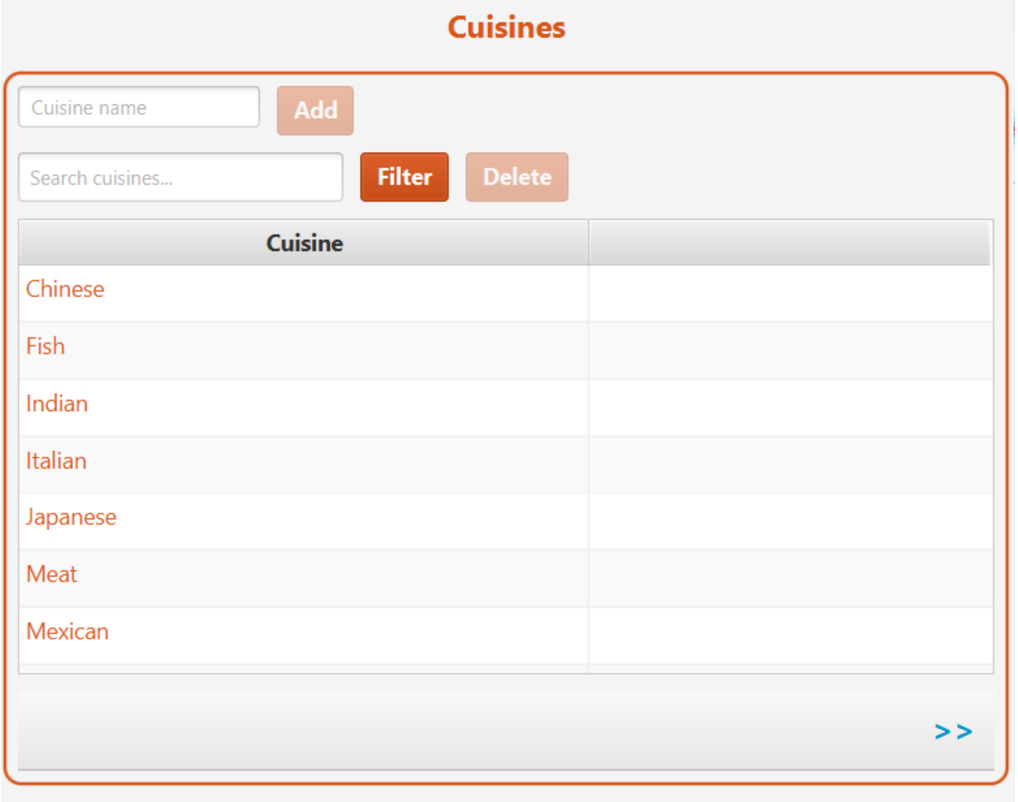
Cities

Name	Latitude	Longitude
Pisa	43.43	10.24
Roma	41.54	12.29
Torino	45.04	7.42
Viareggio	43.86	10.26

Figure 3.30: List of cities after a delete operation.

3.5.4 Manage cuisines

The same operations of cities can be done also for cuisines, so admin can create, modify or delete a cuisine, in the same way wrote on the last subsection (Figure 3.31).



Cuisine	
Chinese	
Fish	
Indian	
Italian	
Japanese	
Meat	
Mexican	

Figure 3.31: Admin cuisines view.