POLITECNICO
MILANO 1863

Iqueue Project

UML Architectural

Software Engineering for Automation (2022-2023)

Giacomelli Gianluca, 10615105
Gottardini Andrea, 10617000
Veronese Niccolò Enrico, 10620278

Professors: Rossi Matteo Giovanni
Lestingi Livia

Summary

UML Architectural helps to understand the architectural class diagram of our Iqueue project. The architectural UML has the goal of describing the relationships among the different actors in terms of the internal software components. This means that the Iqueue app will be divided into modules, each one with its own functionalities and which communicate with other components through interfaces. For this reason, the document is divided into three main sections: first a description of the principal components of the Iqueue system is given, then the main interfaces among modules are explained and finally practical examples are shown with Sequence diagrams.

1. Components

In this section the main components of the system are analysed.

- User GUI

The User GUI represents the graphic interface for the User. It is differentiated from the Customer and Shop Owner GUI since in our app the user can be of these two types and thus he will have a different GUI according to the typology of user (Customer or Shop Owner) he chooses. The User GUI is characterized by the methods User_login, User_logout and User_registration which allow the user to login, logout and register in the Iqueue app respectively. Focusing on the User_registration(in Name:String, in Surname:String, in Birthday:Date, in Email:String, in Password:String), means that the user should give as input its name, surname, birthday, email address and password to register.

- Customer GUI

The Customer GUI represents the graphic interface for the user who has selected to be a customer. It is characterized by the id of the customer, the position and the rewards he has received.

- Shop Owner GUI

The Shop Owner GUI represents the graphic interface for the user who has selected to be a shop owner. It has as attribute the shop owner id.

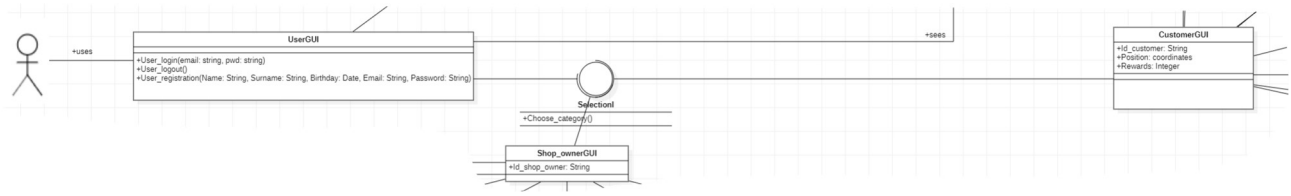

Figure 1: Relationships among User, Shop_owner and Customer GUI.

- Dispatcher_Customer

The dispatcher_customer component has the crucial role of receiving request from the other component interfaces and send them to other interfaces, managing in this way the different tasks. This dispatcher element is specific for the Customer requests and there is a dual component for the Shop Owner. Due to the meaning of this component, it is naturally connected to many interfaces as shown in the following figure.

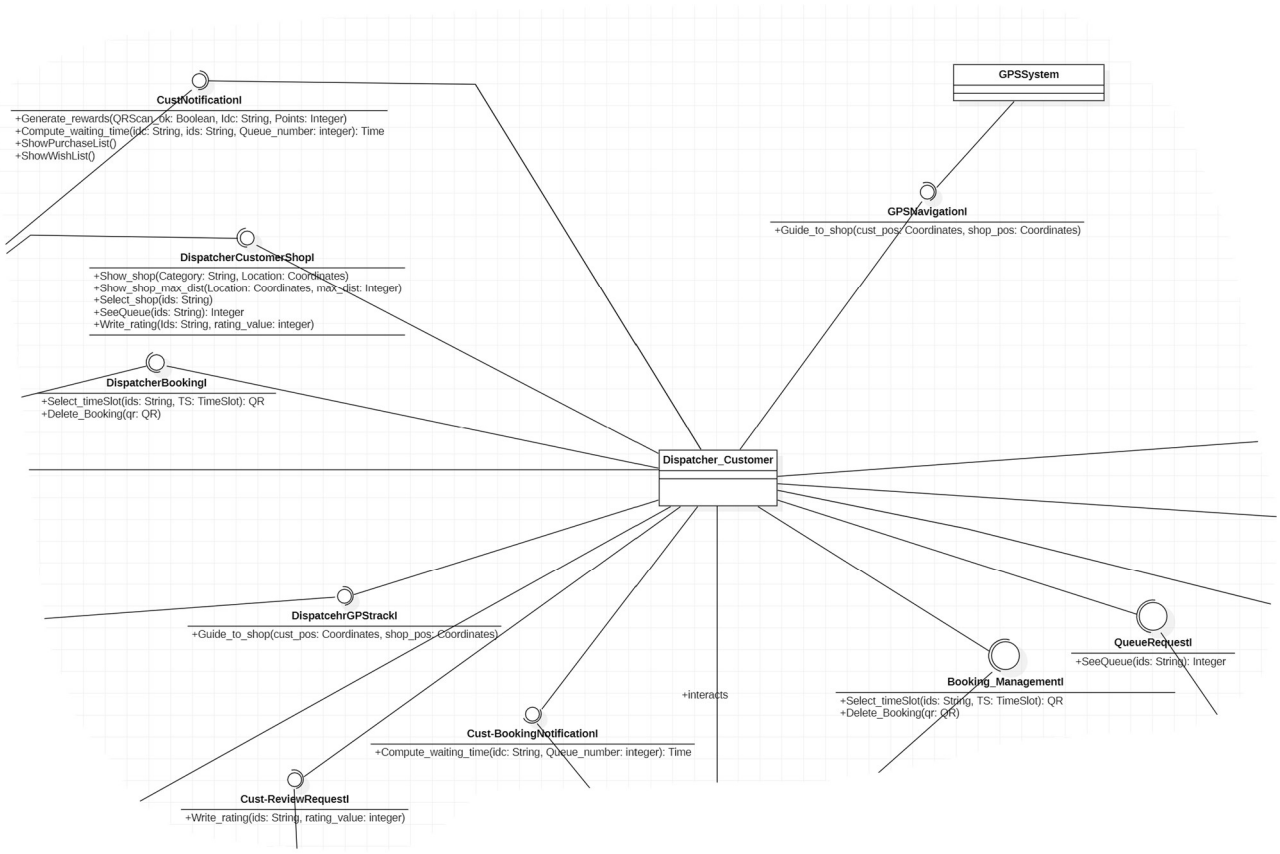

Figure 2: Dispatcher customer relationships

- Booking_manager

The booking manager has the function of supervising the bookings of the time slots from the Customers. For this reason, it is strictly connected to the QRcode manager and the Queue manager, that are the other components used for the booking algorithm.

- QRcode_manager

This component is focusing on the QR management, since the QR code is the fundamental tool with which the Iqueue app increment or decrement the queue and manages the bookings.

- Queue_manager

This module is strongly connected to the previous ones and its role is to coordinate the queue counter coherently with the bookings.

- Review_manager

This component is focusing on the reviews management.

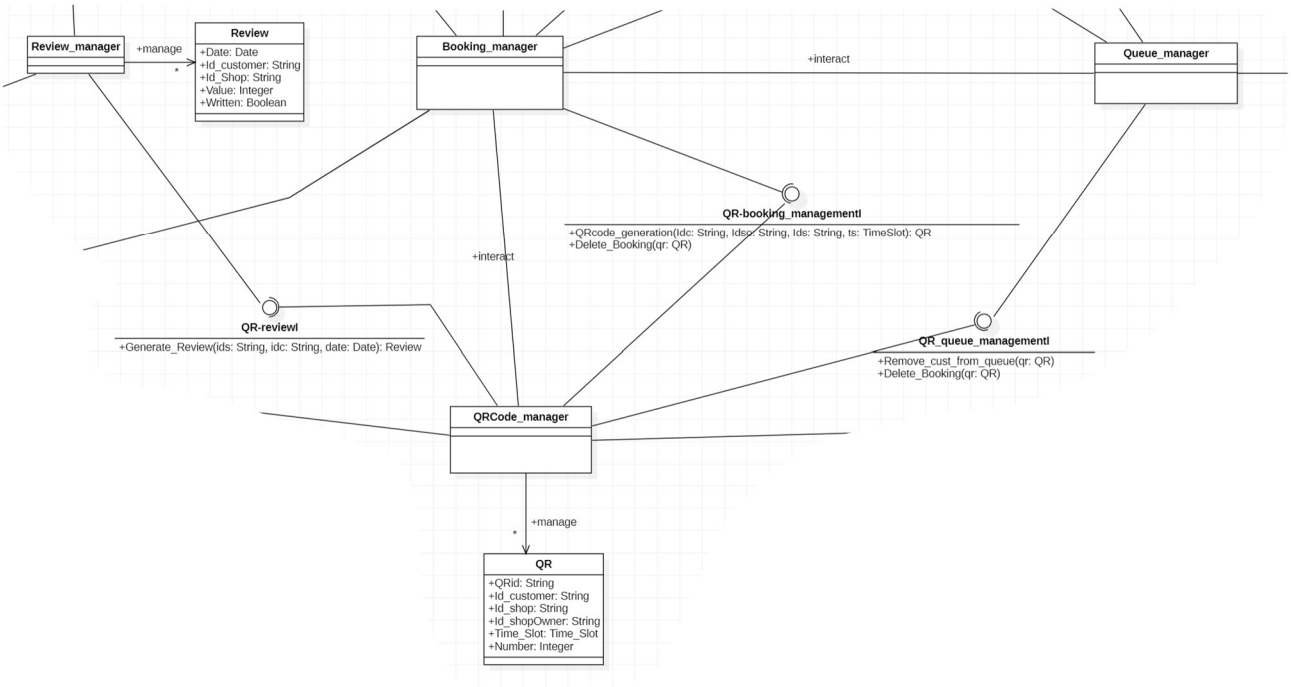

Figure 3: Relationships among Booking_manager, QRcode_manager, Queue_manager and Review_manager.

- Dispatcher_ShopOwner

The dispatcher_shopowner has the same role of the dispatcher_customer, but it is focused on the Shop Owner activities.

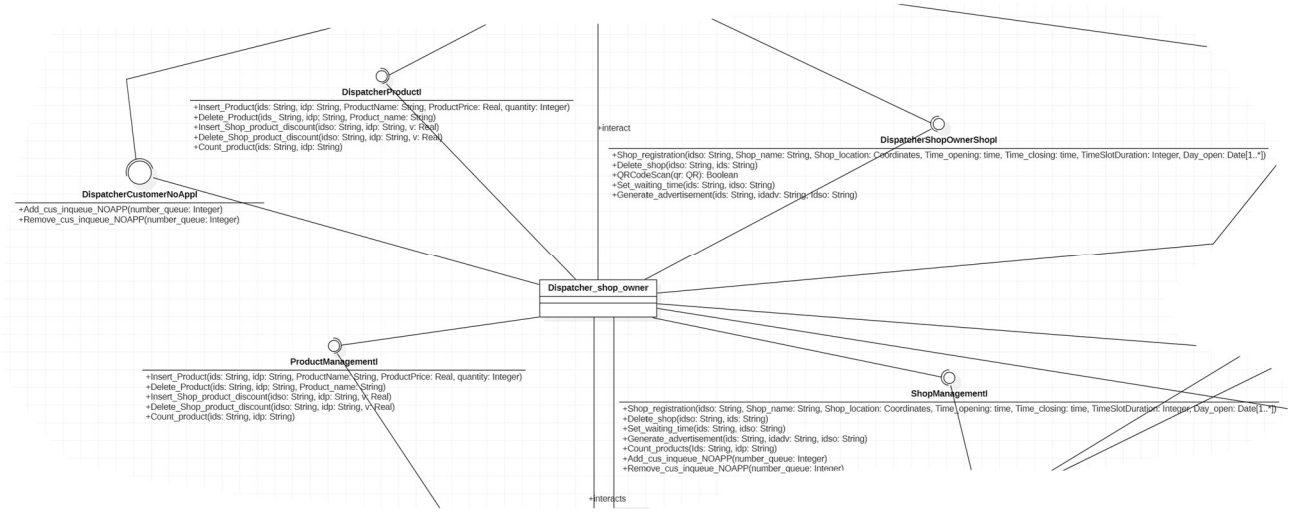

Figure 4: Dispatcher shop owner relationships.

- Shop_manager

The Shop manager is a crucial component since it has functionalities both on Shop Owner side and on Customer side. In fact, on one hand it collects the shop registration information from the Shop Owner, and on the other hand it manages the requests on the shop made by the Customer.

- Product_manager

The Product manager is responsible for the product bookings and hence it is, as the Shop manager, connected both to the Customer and to the Shops.

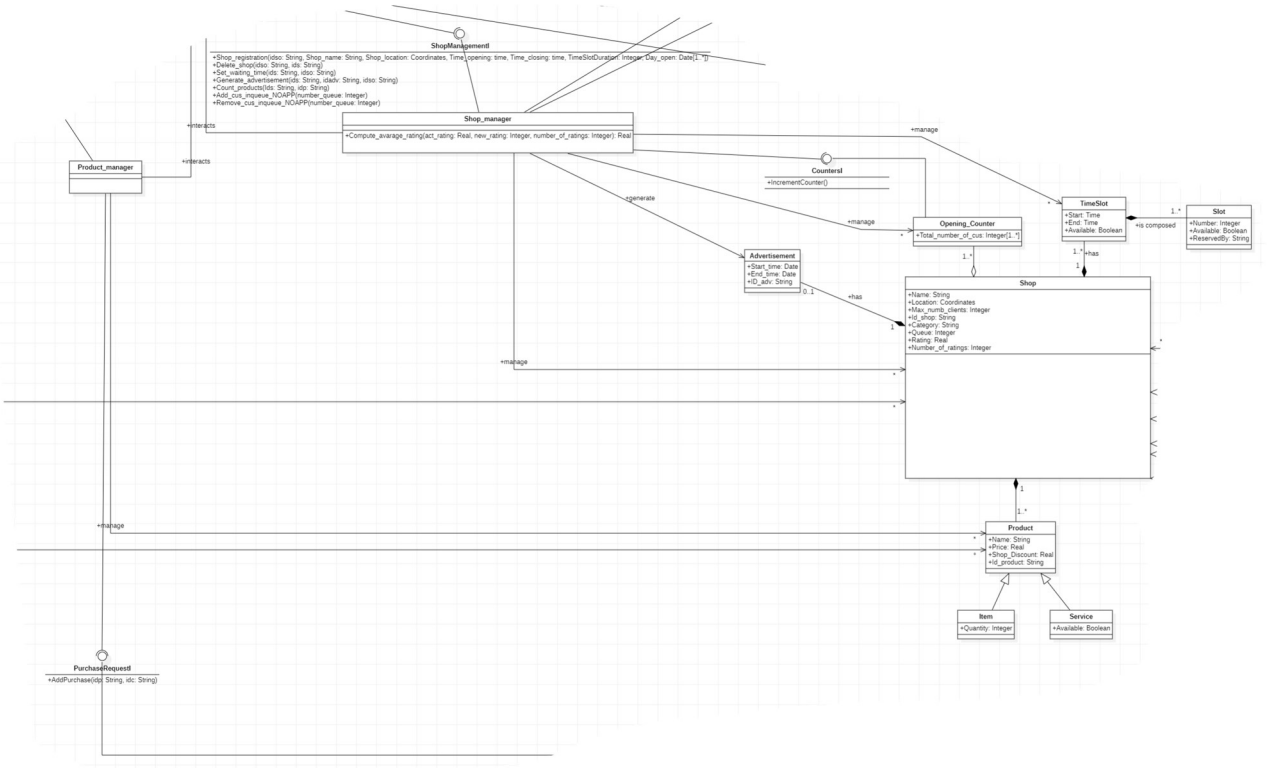

Figure 5: Shop_manager and Product_manager relationship.

- Subscription_manager

It takes care of the User subscriptions in the Iqueue app.

- Gps system

It represents the role of the external GPS system which is used by Iqueue to show the shops to the customers.

2. Interfaces

The interfaces represent the connection and the boundaries between the different components of the system. They are often characterized by operations which may be invoked when passing from one module to the other. They are fundamental since they enable modularity and flexibility in software design. They allow us to define interchangeable components that can be easily swapped or extended without affecting the rest of the codebase, permitting information hiding and design for change. The main interfaces of the Iqueue system are:

- selectionI

The selection interface is characterized by a single method which is Choose_category() and which allows the user to select its role in the Iqueue application: either a Customer or a Shop Owner.

- booking_ManagementI

The booking management interface deals with the booking of the time slots for the Customer and in fact it is characterized by 2 main methods: Select_timeSlot(in ids:String, in TS:TimeSlot): QR which allows to select the corresponding time slot given the id of the shop (ids) and the chosen time slot TS. The output will be a QR code which will be used to confirm the booking once arrived at the Shop. The other method is Delete_Booking(in qr:QR) which allows to delete a previous reservation.

- QueueRequestI

The QueueRequestI allows the Customer, thanks to the method SeeQueue(in ids:String): Integer, to see the number of people in queue in that moment at the Shop.

- Queue-cust_managementI

This interface has one method: Compute_waiting_time(in idc:String, in ids:String, in Queue_number:integer): Time. Thanks to that it is possible to compute the time that a Customer (idc) should wait given the number of people in queue at a Shop (ids) in that moment.

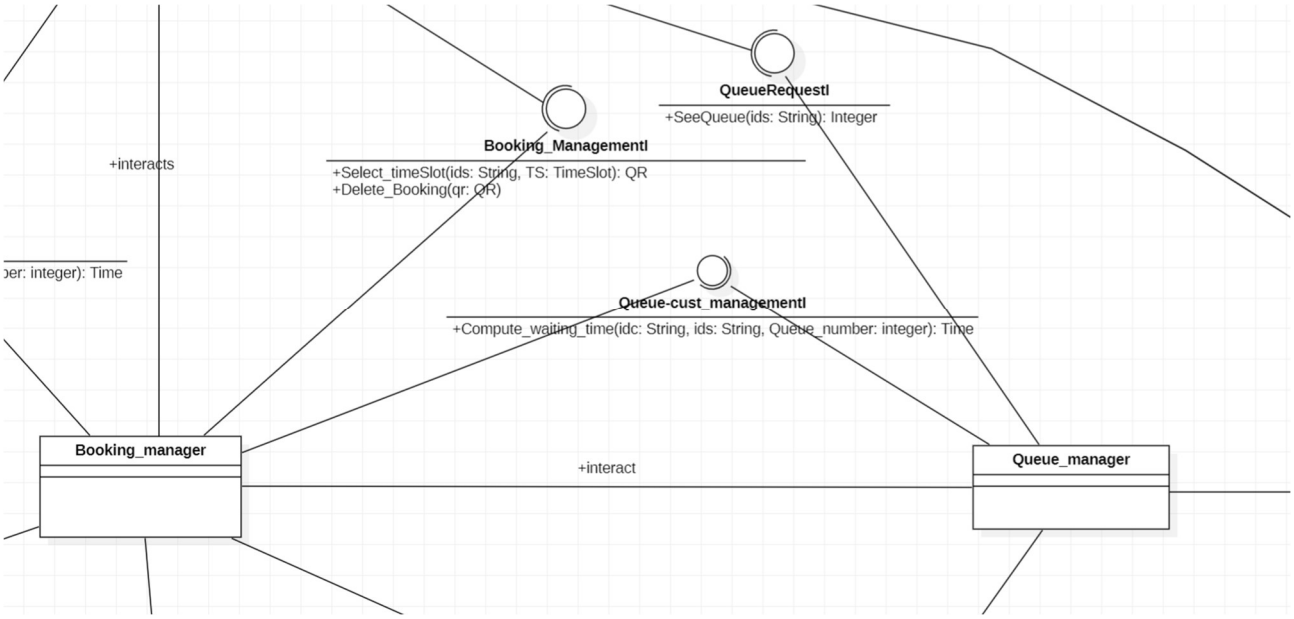

Figure 6: Relationships Booking_ManagementI, QueueRequestI and Queue-cust_managementI

- GPSNavigationI

The GPSNavigation interface, through the Guide_to_shop(in cust_pos:Coordinates, in shop_pos:Coordinates) method allows to guide the Customer to the selected Shop.

- ShopManagementI

The Shop management interface is characterized by many methods like Shop_registration(in idso:String, in Shop_name:String, in Shop_location:Coordinates, in Time_opening:time, in Time_closing:time, in Day_open:Date[1..*]), useful to register a Shop in the Iqueue app, or Generate_advertisement(in ids:String, in idadv:String, in idso:String) which allows to generate an advertisement for the Shop in order to have possible financial gains. Two other important methods in this interface are Add_cus_inqueue_NOAPP(in number_queue:Integer) and

Remove_cus_inqueue_NOAPP(in number_queue:Integer) which represent the manual increment or decrement of the number of people in queue that the Shop Owner should do when a person without the Iqueue app enters in the Shop. Other methods guarantee also to remove the shop or to compute the average rating that the Shop has, based on the reviews of the Customers.

- ProductManagementI

This interface is characterized by some methods and its main functionality is to tackle the problem of adding a product linked with a Shop, together with its corresponding discount so that to guarantee that a Customer using the Iqueue app can have economic benefits. In order to deal with the above theme, the methods are Insert_Product(in ids:String, in idp:String, in ProductName:String, in ProductPrice:Real, in quantity:Integer) and Insert_Shop_product_discount(in idso:String, in idp:String, in v:Real). Finally Count_product(in ids:String, in idp:String) allows the Shop Owner to keep track of the number of products in his Shop so that to generate some statical analysis on them.

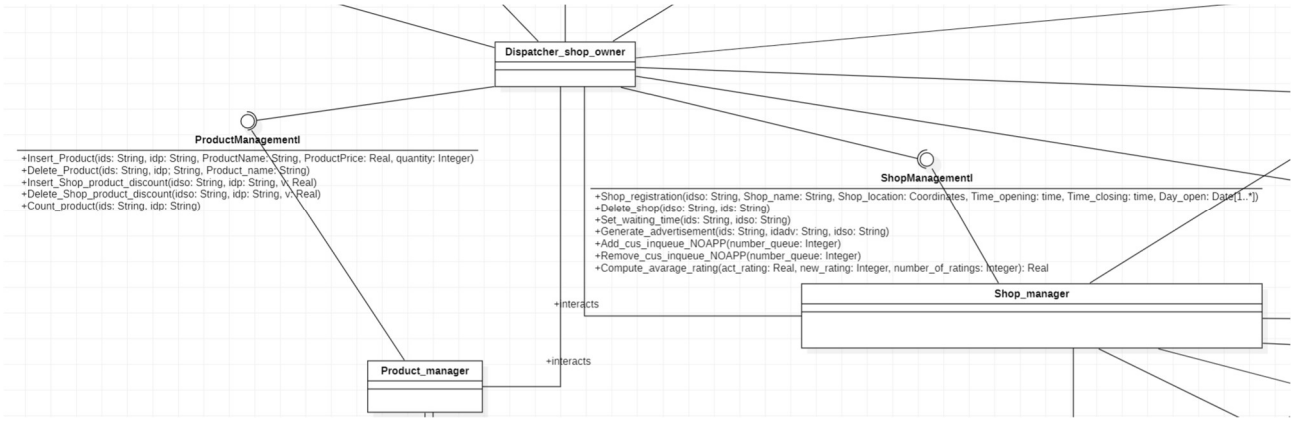

Figure 7: Relationships between ShopManagementI and ProductManagementI.

- Cust-shopRequestI

This interface is characterized by some methods which link the Customer with the Shop like Show_shop(in Category:String, in Location:Coordinates) that allows to see on the map the shops of a given category. Then with Select_shop(in ids:String) it is possible to select a specific shop among the one shown and with Show_shop_max_dist(in Location:Coordinates, in max_dist:Integer) the Customer can see the maximum distance from the Shop.

- Cust-ReviewRequestI

This interface permits the interaction between a customer and his reviews, in particular it allows to write a rating for a Shop using the method Write_rating(in ids:String, in rating_value:integer).

- Review-ShopI

This interface permits to propagate the effect of the previous cited Write_rating(in ids:String, in rating_value:integer) to the relative Shop.

- QR-reviewI

This interface grant the generation of the review immediately after that a shop owner has scanned the customer reservation QR by means of Generate_Review(in ids:String, in idc:String, in date:Date): Review.

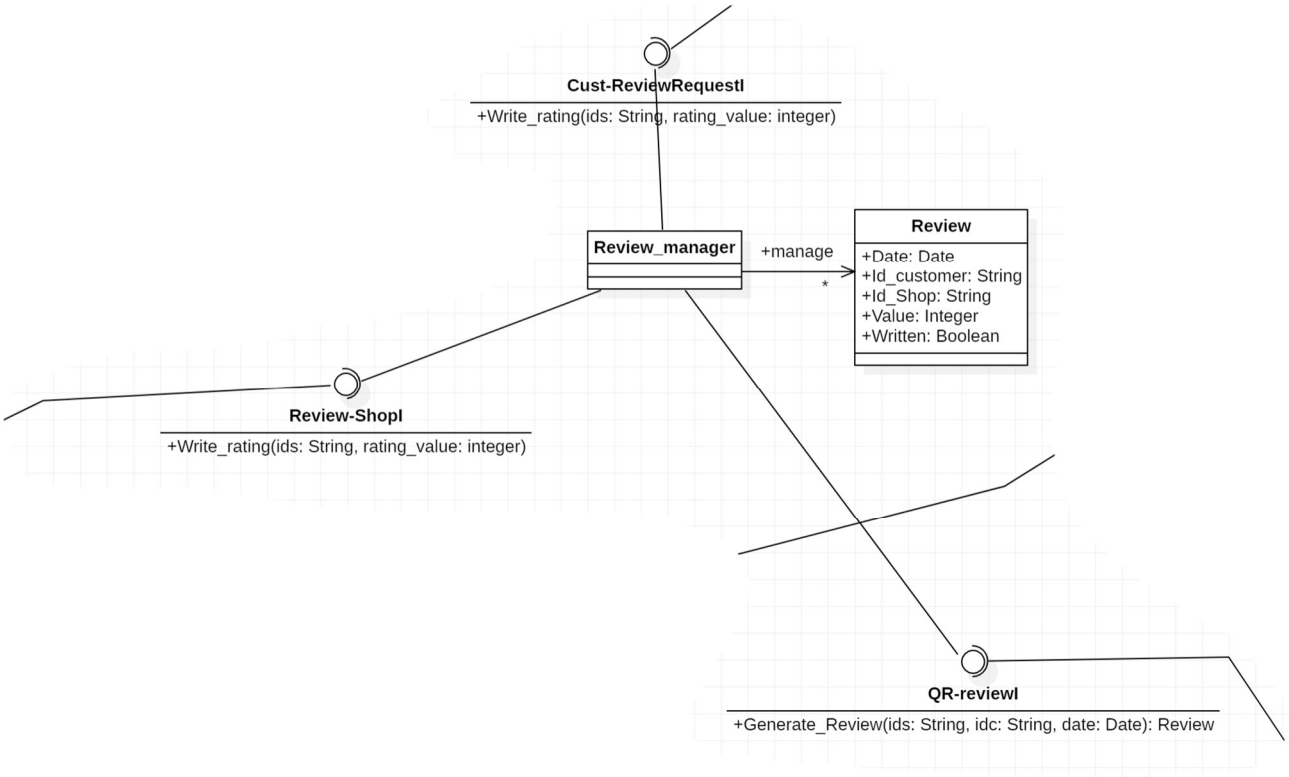

Figure 8: Relationships among Cust-ReviewRequestI, QR-reviewI and Review-ShopI

- QR-booking_managementI

This interface allows to generate a QR code every time a booking from the Customer (idc) in the Shop (ids) with the Shop Owner(idso) and in the corresponding time slot ts. QRcode_generation(in Idc:String, in Idso:String, in Ids:String, in ts:TimeSlot): QR

- QR_queue_managementI

This interface guarantees that every time a QR code has been scanned the corresponding Customer is removed from the Queue, thanks to the method Remove_cust_from_queue(in qr:QR)

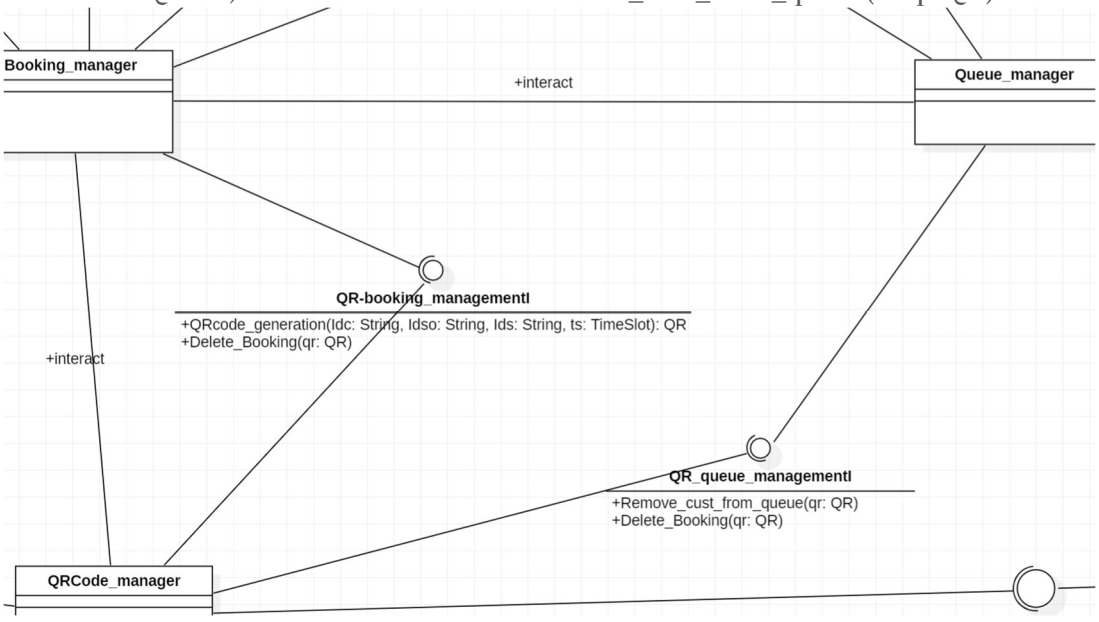

Figure 9: Relationships among QR-booking_managementI and QR_queue_managementI

- Booking-ShopOwnI

It sends a notification to the Shop Owner every time a booking is created. Sent_notification(in idso:String)

- Wish List RequestI

This interface deals with the Wish List feature of the Iqueue app. Thanks to Add_to_wishlist(in idp:String, in idc:String) it is possible to add product (idp) in the corresponding whishlist of the Customer. The dual method is Remove_from_wishlist(in idp:String, in idc:string) and finally with ShowWishList() the Customer may see its whole whishlist.

- PurchaseRequestI

It is characterized by the method Show_PurchaseList() which allows the Customer to see its purchase list of products.

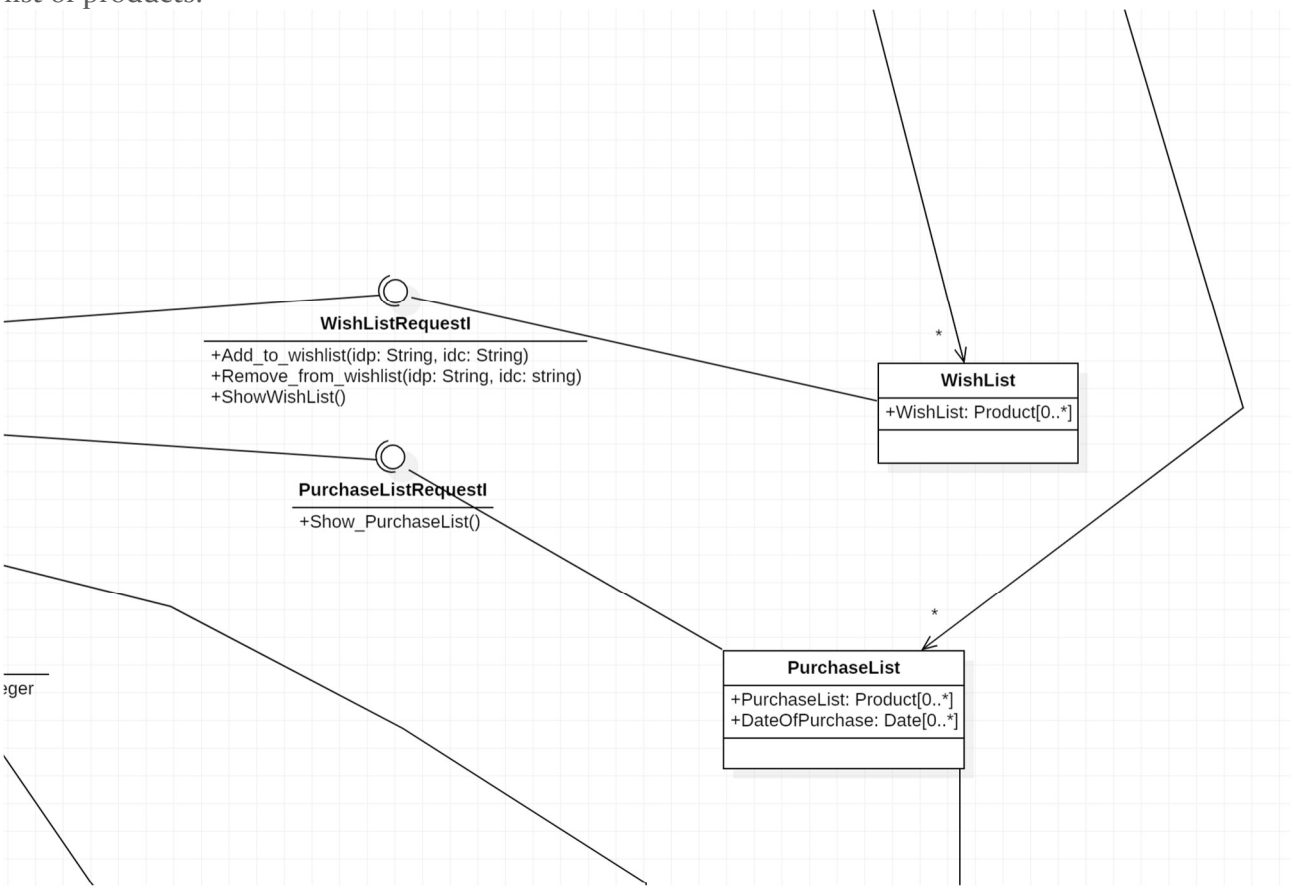

Figure 10: Relationships among Wish List RequestI and PurchaseListRequestI

- CountersI

This interface is related to the tracking of the total number of Customer in a Shop in a certain period, so that to give the Shop Owner the possibility of developing some analysis on his Shop. IncrementCounter()

- QR-Shop_owner_managementI

This interface thanks to the method QRCodeScan(in qr:QR): Boolean provides the Shop Owner the possibility to scan the QR of the Customer who has booked with the Iqueue app.

- Dispatcher interfaces

Regarding the dispatcher interfaces, they are characterized by the same methods of the above interfaces and they are connected to the dispatcher elements for the Customer or for the Shop owner. In this way, the architecture UML is separated in more modules and components so that to guarantee an higher level of comprehension.

3. Sequence diagrams

In this section the sequence diagrams related to the use cases described in the RASD document are presented (see RASD section 3.2). These diagrams help to understand the relationships between the different components of the system and they represent the dynamic behaviour of the Iqueue app processes. The sequence diagrams can be found in the file UML ARCHITECTURAL.mdj, accessing them inside the UML Class Diagram through the Model Explorer. To simplify the reading, the description of the use cases is present.

Use Case name: Book

- Participating actors: Client, Shop Owner, Iqueue, GPS system
- Flow of Events:
  1. The Client selects a certain type of shops (bakeries in the scenario).
  2. Iqueue show on his map a list of the available shops of the indicated type with their queues.
  3. The Client selects a shop.
  4. Iqueue shows the available products, time slots and special offers.
  5. The Client selects a free time slot.
  6. A QR is generated corresponding to the booking
  7. Iqueue notifies the Shop Owner of the time slot booking.
  8. The Client chooses to be guided to the shop.
  9. Iqueue redirects the Client to a third-party GPS system and provides to it the shop address.

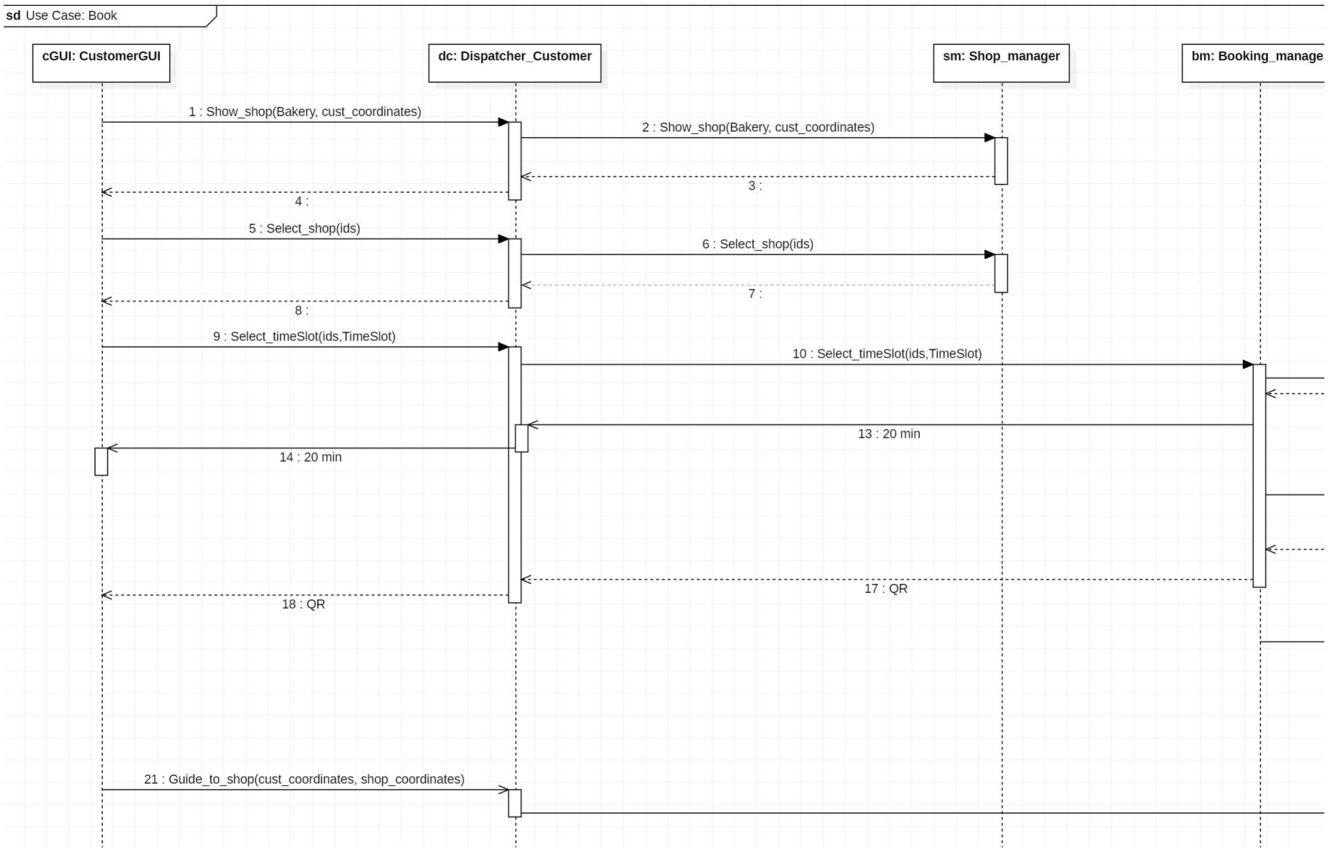

Figure 11: Sequence diagram book PART 1

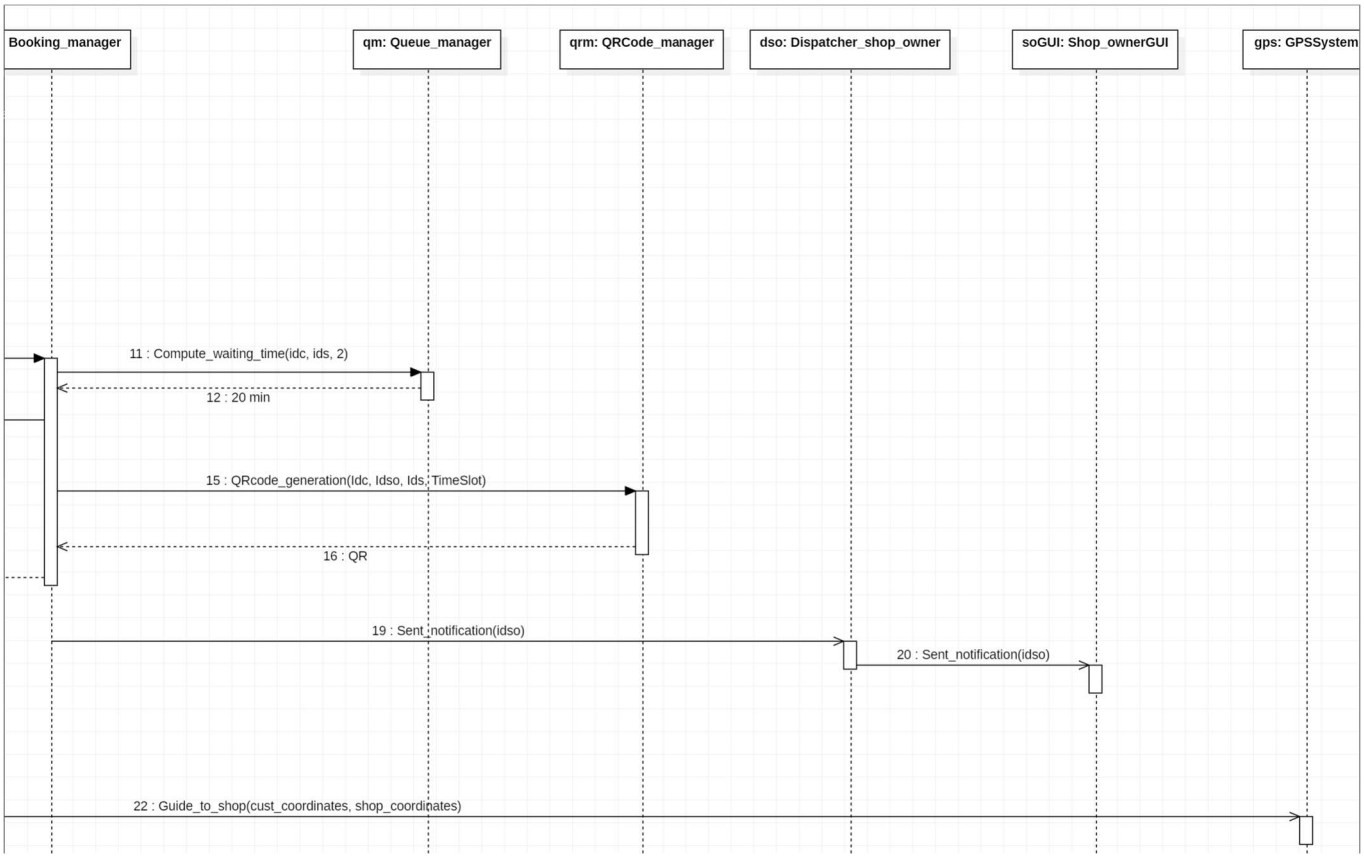

Figure 12: Sequence diagram Book PART 2

Use Case name: Count clients

- Participating actors: Client, Shop Owner, Iqueue
- Entry Condition: Client enters in a shop.
- Flow of Events:
  1. Shop Owner asks the Client if he has done a reservation with Iqueue.
  2. The Shop Owner scans the client QR code.
  3. Iqueue increments the counter client for the shop of the relative shop opening.

- Exit Condition: The use case terminates after the increment of the client counter. Shop Owner can consult the client counter value on different shop opening accessing Iqueue.
- Exceptions: If the Client has not done his registration on Iqueue, the Shop Owner increments the client counter on Iqueue by himself.

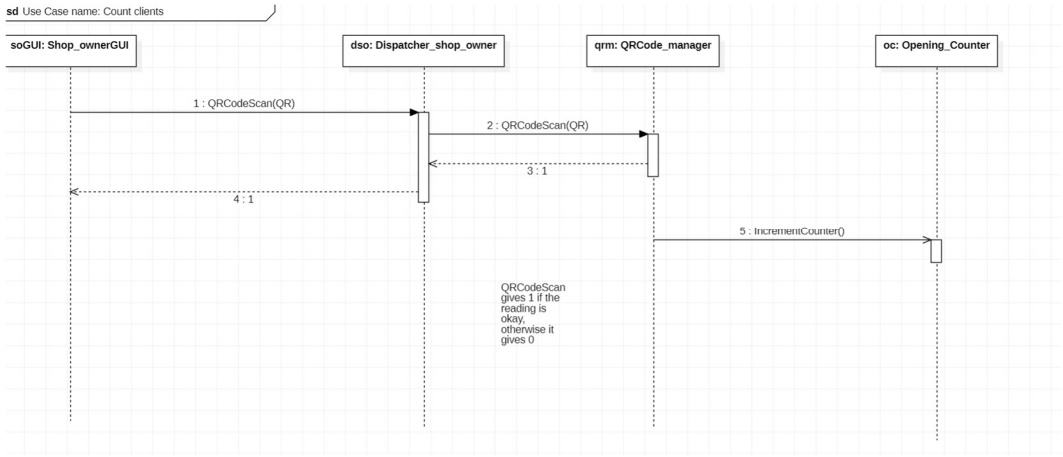

Figure 13: Count clients sequence diagrams

Use Case name: Register Shop

- Participating actors: Shop Owner, Iqueue
- Entry Condition: Shop Owner has a shop.
- Flow of Events:
  1. Shop Owner starts register Shop procedure on Iqueue.
  2. Iqueue asks shop position to the Shop Owner.
  3. Shop Owner inserts the shop position.
  4. Iqueue asks shop logo and/or pictures to the Shop Owner.
  5. Shop Owner inserts the shop logo and/or pictures (optional).
  6. Iqueue asks Shop Owner to insert products of his shop.
  7. Shop Owner inserts the products.
  8. Iqueue asks Shop Owner to insert special offers on products.
  9. Shop Owner inserts the special offers on products (optional).
- Exit Condition: The use case terminates after the product discounts entering. Shop registration is now terminated, and the shop is now visible on the Iqueue map with its products.

- Exceptions: The insertion of shop position and products is mandatory to conclude the shop registration on Iqueue; The Shop Owner could not insert immediately pictures, the logo of his shop and the special offers on products to register his shop. They could be inserted in a second moment.

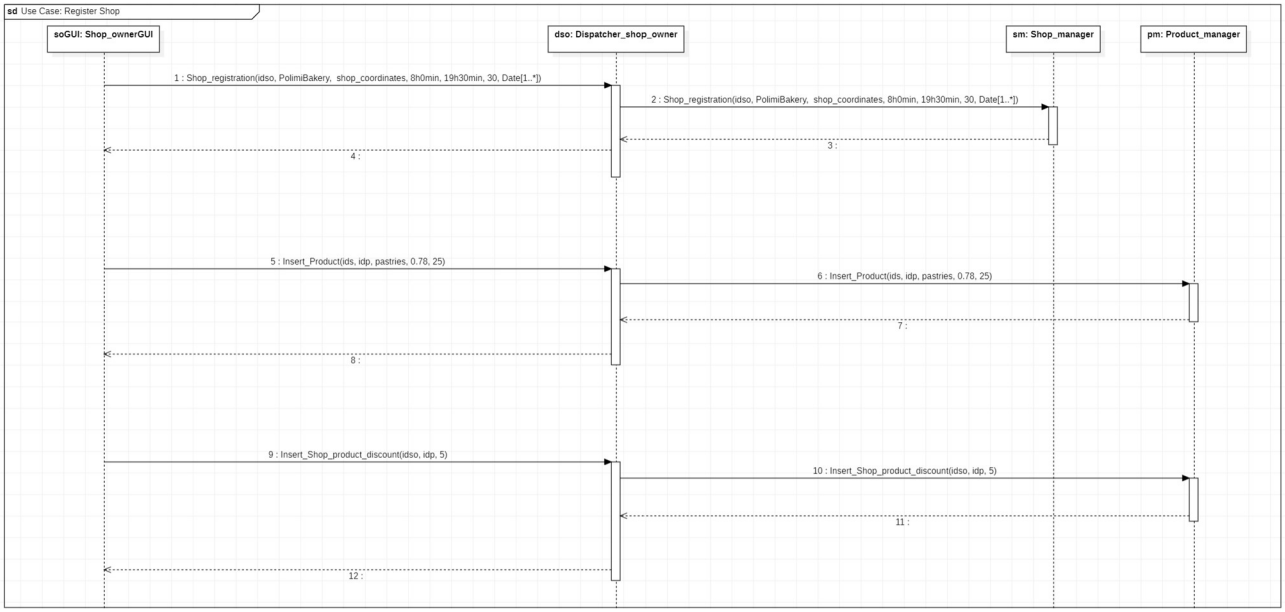

Figure 14: Register shop sequence diagram.

Use Case name: Search

- Participating actors: Client, Iqueue
- Entry Condition: Client likes to know shops in his surroundings.
- Flow of Events:
  1. Client allows Iqueue to know his position.
  2. Client selects a certain type of shops.
  3. Iqueue show on his map a list of the available shops of the indicated type with their queues.
  4. Client selects a maximum distance from the shops.
  5. Iqueue shows the shops accordingly with the selected distance.
  6. The Client selects a shop.
  7. Iqueue shows the available products and special offers.
- Exit Condition: The use case terminates after the Iqueue shows the products and special offers. Client could now book a time slot or simply exit the application.

- Special Requirements: The Client must have a device which is able to provide the position to the Iqueue.

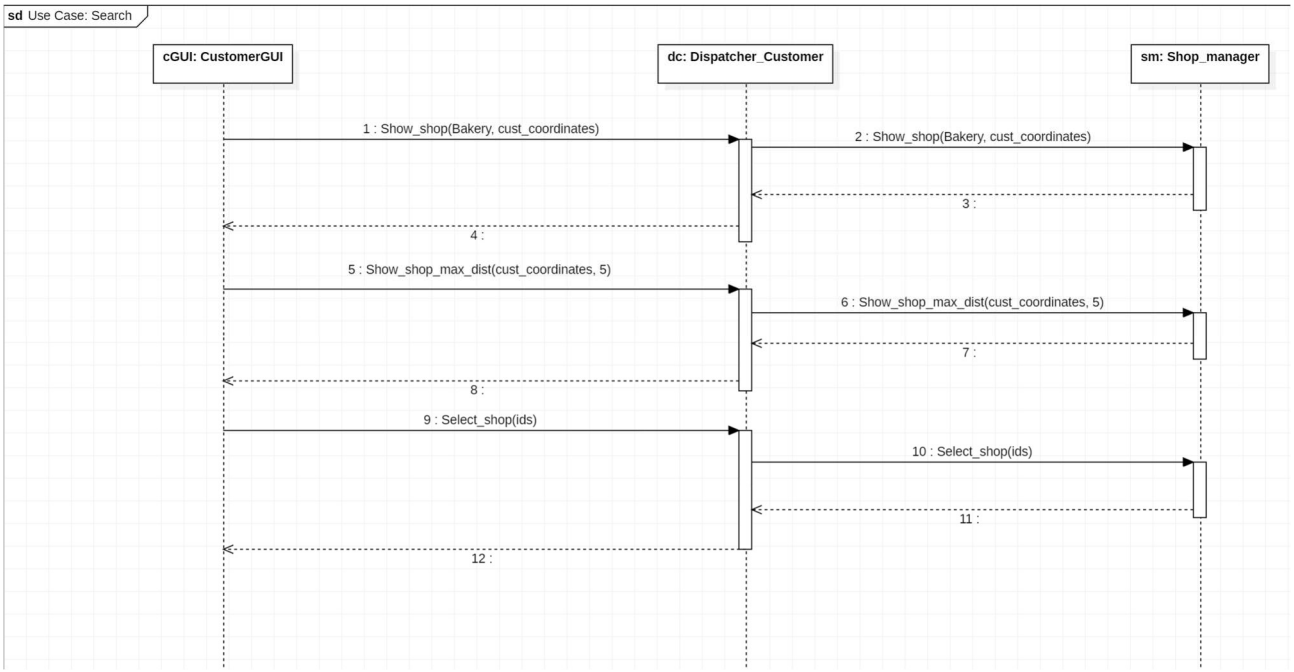

Figure 15: Search sequence diagram.