ENJOY RESERVATION

SERVICE ORIENTED SOFTWARE ENGINEERING

Lecturer: Prof. Marco Autili

Team Members: Luca Grillo, Aysel Yusubzada, Cristian Capozucco, Davide Mariotti

TABLE OF CONTENTS

Table of Contents 1
Main Goals 2
UML Diagrams 3
- Component diagram 3
- Sequence Diagrams 6
  - Login 6
  - Logout 7
  - Check Session 8
  - Check Owner 9
  - Restaurant Inserting 10
  - Restaurant Information 11
  - Restaurant Single Information 12
  - Restaurant Update 13
  - Restaurant Delete 14
  - Restaurant Booking 15
  - Cinema Inserting 16
  - Cinema Information 17
  - Cinema Single Information 18
  - Cinema Update 19
  - Cinema Delete 20
  - Cinema Booking 21
  - Night Information 22

Installation 23
- Technologies and Tools 23

API REST 24
- Account Control 24
- Cinema Controller 25
- Restaurant Controller 28
- Night Controller 30

Mobile Application 31

MAIN GOALS

Our project regards a SOA-based application that allows users to get a reservation in a restaurant (for a table or the whole place) and/or in a cinema. The restaurants and the cinemas must be enrolled in the system through the use of a service. So, the users can get a reservation only in restaurants and in cinemas that have an agreement with the service. Users can view the restaurants and the cinemas nearest to them thanks to a service provided by an external provider (Google). If users get reservation for both restaurant and cinema, they may get a discount code for the cinema. The restaurants are divided in categories (for example different styles and cuisines), and the users can see their menu afterwards. Users can see which films are provided by cinemas and can compare the prices of the cinemas (the prices can be different for each cinema and different time of the week or day).

UML DIAGRAMS

The following section describes the architecture of the system. A Component Diagram and some Sequence Diagrams are shown below.

COMPONENT DIAGRAM

The following list explains all the components in the architecture we build.

- **Enjoy Reservation App (client)**: This component represents our Mobile App build with Ionic, Cordova and AngularJS.

- **RoutingRequest**: This component provides the direct back-end communication for the front-end application. It represents a comprehensive and reusable REST API back-end built with Spring. The "RoutingRequest" component provides an interface to the "EnjoyReservationApp" component. We added all the URL to access the services in the next section.

- **EnjoyReservation**: This component represents the prosumer of our application. It allows the interaction with all the services of our system: Restaurant Inserting service, Restaurant Information service, Restaurant Booking service, Cinema Inserting service, Cinema Information service, Cinema Booking service, Account Manager, Google Maps service (external service). The communication with the services can be parallel, two services at the same time or one service at time (as you will see in the dynamic view of our system in the next pages). It also provides interface to the "RoutingRequest" component.

- **RestaurantBooking**: This component represents the service that allows users to create a booking to a specific restaurant. The "RestaurantBooking" component provides an interface to the "EnjoyReservation" component.

- **RestaurantInformation**: This component represents the service that allows clients to take information about all the restaurants in a specific city. It also provides a function to show all the information about a single restaurant. The "RestaurantInformation" component provides an interface to the "EnjoyReservation" component.

- **RestaurantInserting**: This component represents the service that allows us to insert a restaurant inside our "EnjoyReservation" system and Database. "RestaurantInserting" component also provides function to update and delete a restaurant inside our system. Note that this small does not check the permission (for example everyone can change a restaurant even if they are not the owners). The permission will be checked from "EnjoyReservation" component through

the“AccountManager” component. The “RestaurantInserting” component provides an interface to the “EnjoyReservation” component.

- **CinemaBooking**: This component represents the service that allows users to create a booking to a specific cinema. The “CinemaBooking” component provides an interface to the “EnjoyReservation” component.

- **CinemaInformation**: This component represents the service that allows clients to take information about all the cinemas in a specific city. It also provides a function to show all the information about a single cinema. The “CinemaInformation” component provides an interface to the “EnjoyReservation” component.

- **CinemaInserting**: This component represents the service that allows us to insert a cinema inside the “EnjoyReservation” system and DB. “CinemaInserting” component also provides function to update and delete a cinema inside our system. Note that this small does not check the permission (for example everyone can change a cinema even if they are not the owners). The permission will be checked from “EnjoyReservation” component through “AccountManager” component. The “CinemaInserting” component provides an interface to the “EnjoyReservation” component.

- **GoogleMaps**: This component allows us to retrieve the cinema/restaurant coordinates (latitude and longitude) given its address and the city where it is located. These coordinates will be then stored with the other cinema/restaurant information. The component “GoogleMaps” provides an interface to the “EnjoyReservation” component.

- **DB**: This component represents the Database that contains the information concerning users, cinema and restaurants. The component “DB” provides an interface to the “RestaurantInserting”, “RestaurantInformation”, “RestaurantBooking”, “CinemaInserting”, “CinemaInformation”, “CinemaBooking” and “AccountManager” components.

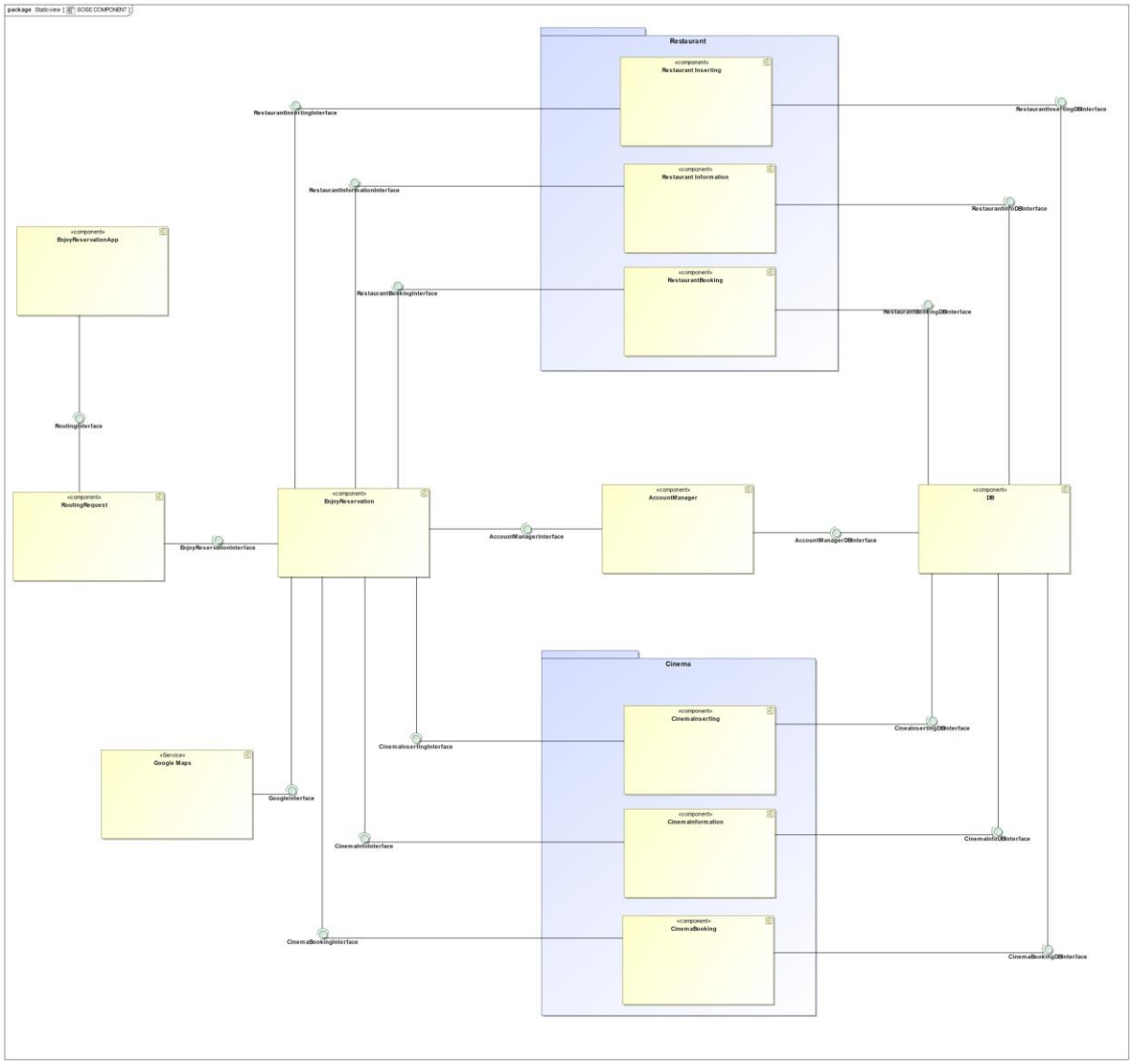

Figure 1. Component diagram of the Enjoy Reservation system

The following list explains all the sequence diagrams we build.

LOGIN

The figure 2 which is given below describes the Login service in the Enjoy Reservation system. User uses his credentials to get access. EnjoyReservationApp sends a JSON containing user's email and password to the RoutingRequest by a REST API via a POST request. The Login request will be forwarded to the the EnjoyReservation component and subsequently to the AccountManager component that checks if the credentials inserted by the user are correct. A response containing the session token and the user_id (if the login was success) is sent to EnjoyReservationApp.

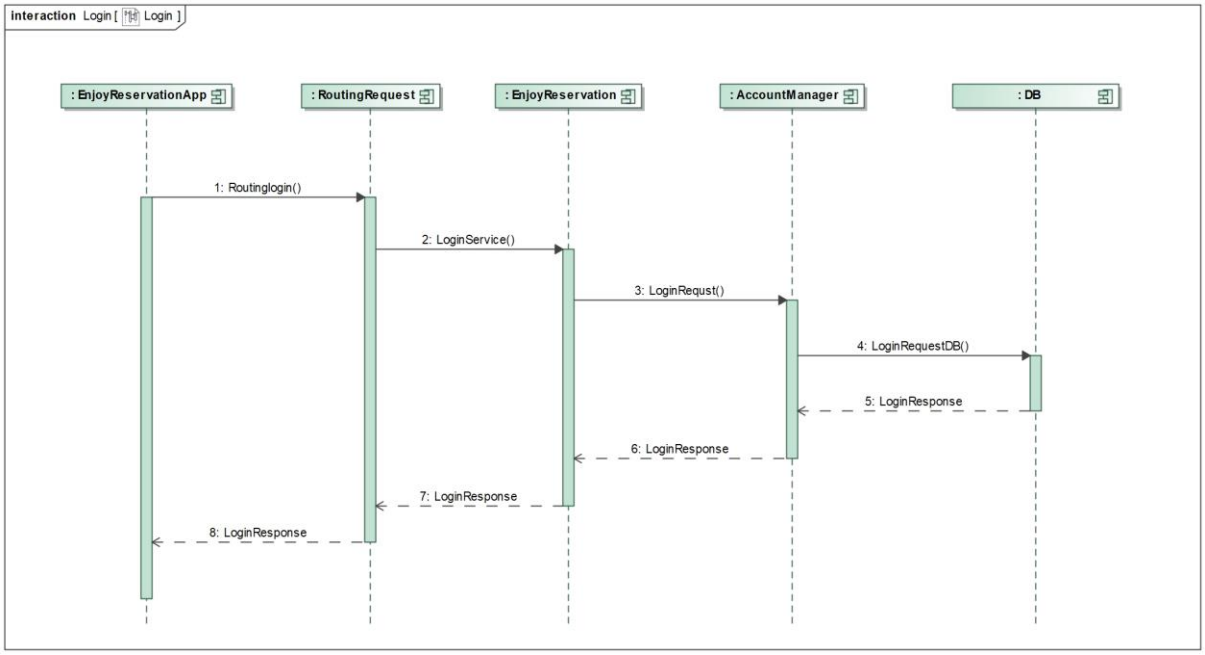

Figure 2. Sequence diagram of the Login service in the Enjoy Reservation system

LOGOUT

The figure 3 which is given below describes the Logout service in the Enjoy Reservation system. User uses this function to logging out. EnjoyReservationApp sends the session token to the RoutingRequest by a REST API. The Login request will be forwarded to the the EnjoyReservation component and subsequently to the AccountManager component that will deal with to delete the session token and complete the logout request. A response containing a boolean answer is sent to EnjoyReservationApp.

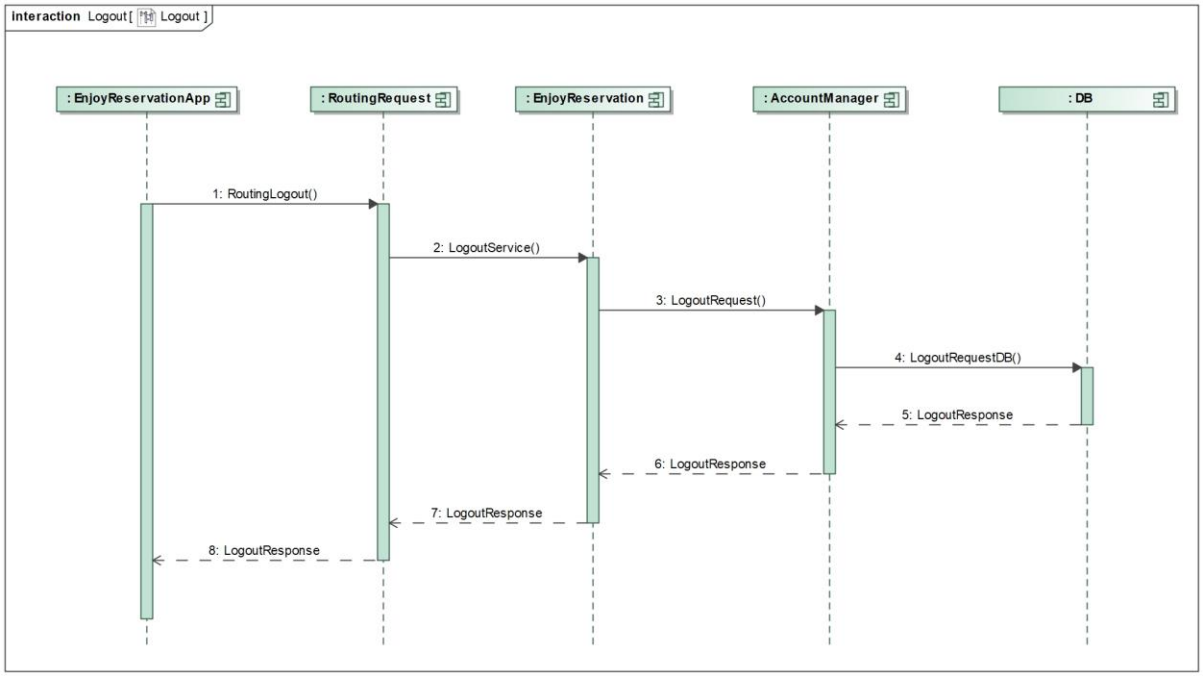

Figure 3. Sequence diagram of the Logout service in the Enjoy Reservation system

CHECK SESSION

The figure 4 which is given below describes the Check Session service in the Enjoy Reservation system. EnjoyReservationApp sends the session token to the RoutingRequest by a REST API. The request will be forwarded to the the EnjoyReservation component and subsequently to the AccountManager component that will deal with to check if the session token is valid and if it is associated to the right user. A response containing a boolean answer is sent to EnjoyReservationApp.

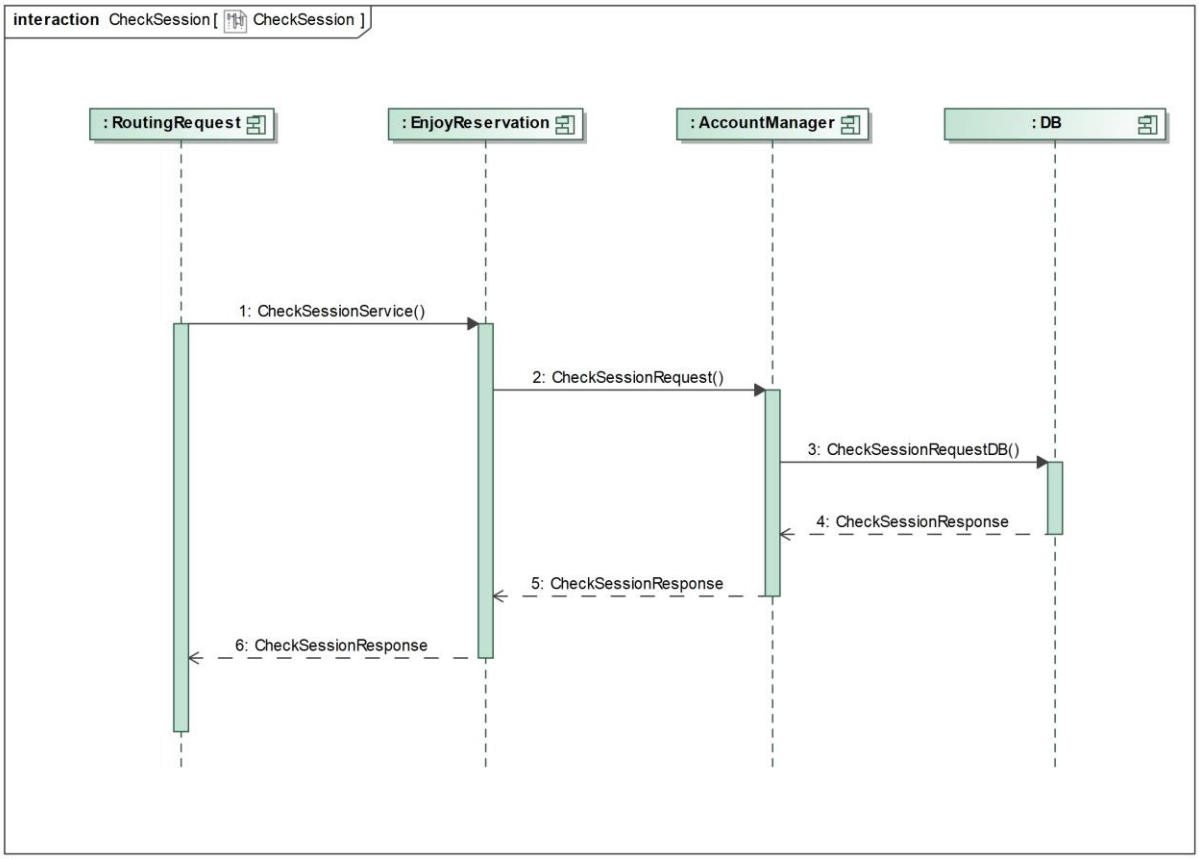

Figure 4. Sequence diagram of the Check Session service in the Enjoy Reservation system

CHECK OWNER

The figure 5 which is given below describes the Check Owner service in the Enjoy Reservation system. EnjoyReservationApp sends a JSON containing the id of the cinema/restaurant to the RoutingRequest by a REST API via a POST request. First of all, the session is checked. If the user is logged, the request will be forwarded to the EnjoyReservation component and subsequently to the AccountManager component that will deal with to check if the user who send the request is the owner of that restaurant/cinema. A response containing a boolean answer is sent to EnjoyReservationApp.

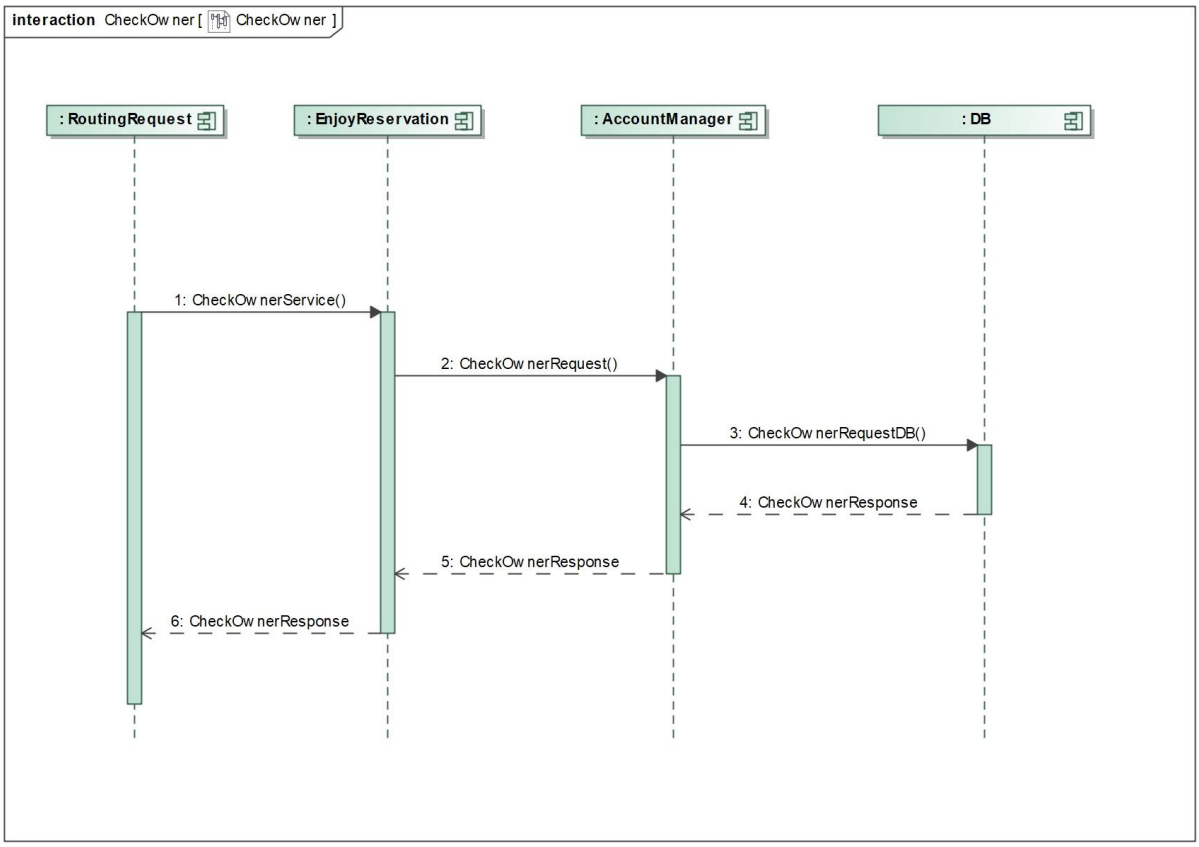

Figure 5. Sequence diagram of the Check Owner service in the Enjoy Reservation system

The figure 6 which is given below describes the Restaurant Inserting service in the Enjoy Reservation system. EnjoyReservationApp sends a JSON containing the information about the restaurant that the owner wants to roll in the system to the RoutingRequest by a REST API via a POST request. First of all, the session is checked. If the user is logged, the request will be forwarded to the the EnjoyReservation component and subsequently to the RestaurantInserting component that will deal with to insert the restaurant on the Database. A response containing a boolean answer is sent to EnjoyReservationApp.

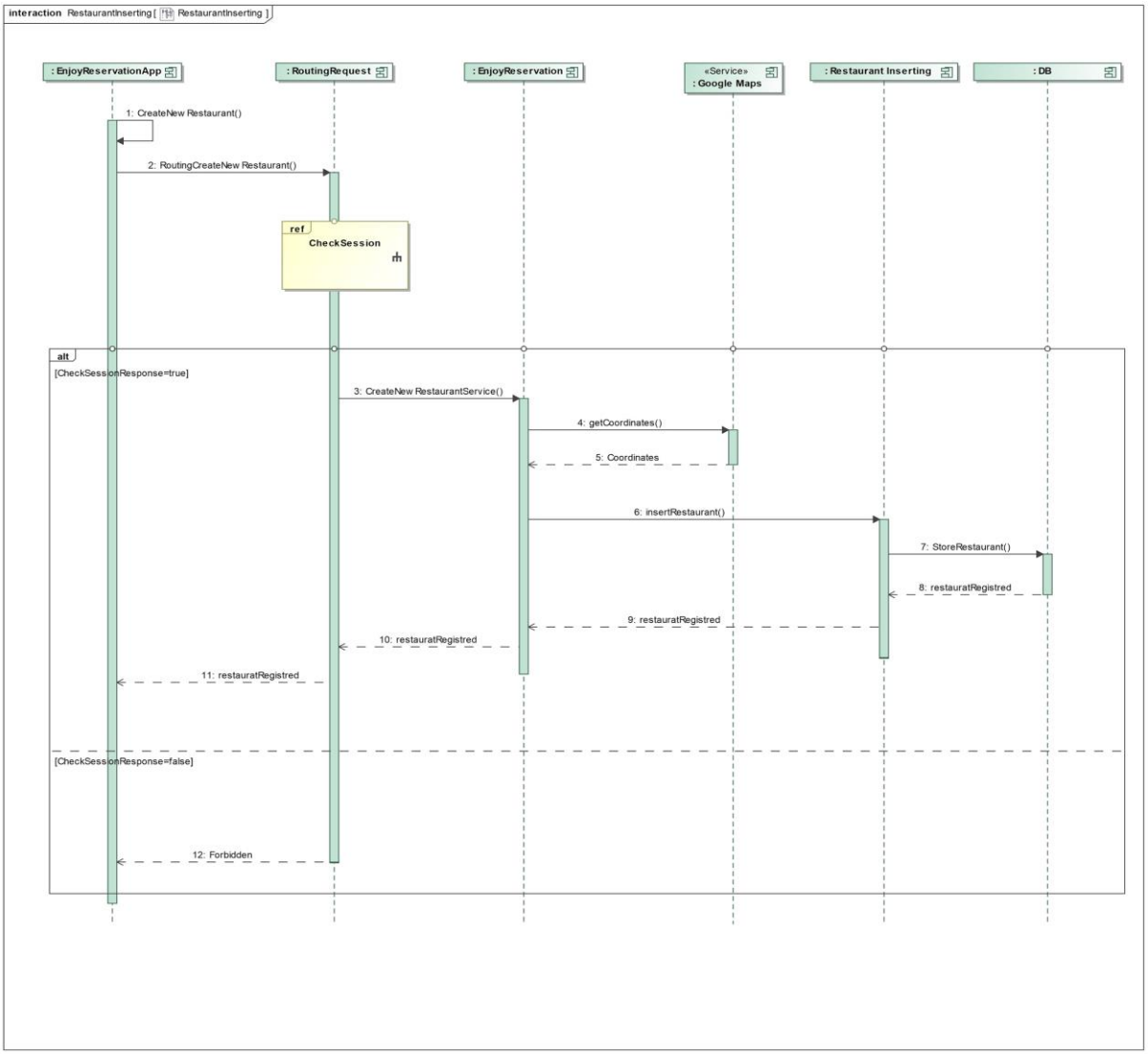

Figure 6. Sequence diagram of the Restaurant Inserting service in the Enjoy Reservation system

The figure 7 which is given below describes the Restaurant Information service in the Enjoy Reservation system. The user selects the city where he wants to see the restaurants information. EnjoyReservationApp sends the city by a REST API via a GET request. First of all, the session is checked. If the user is logged, the request will be forwarded to the EnjoyReservation component and subsequently to the RestaurantIinformation component that will query the Database in order to get all the restaurants in a precise city. A response containing a list of restaurants is sent to EnjoyReservationApp.

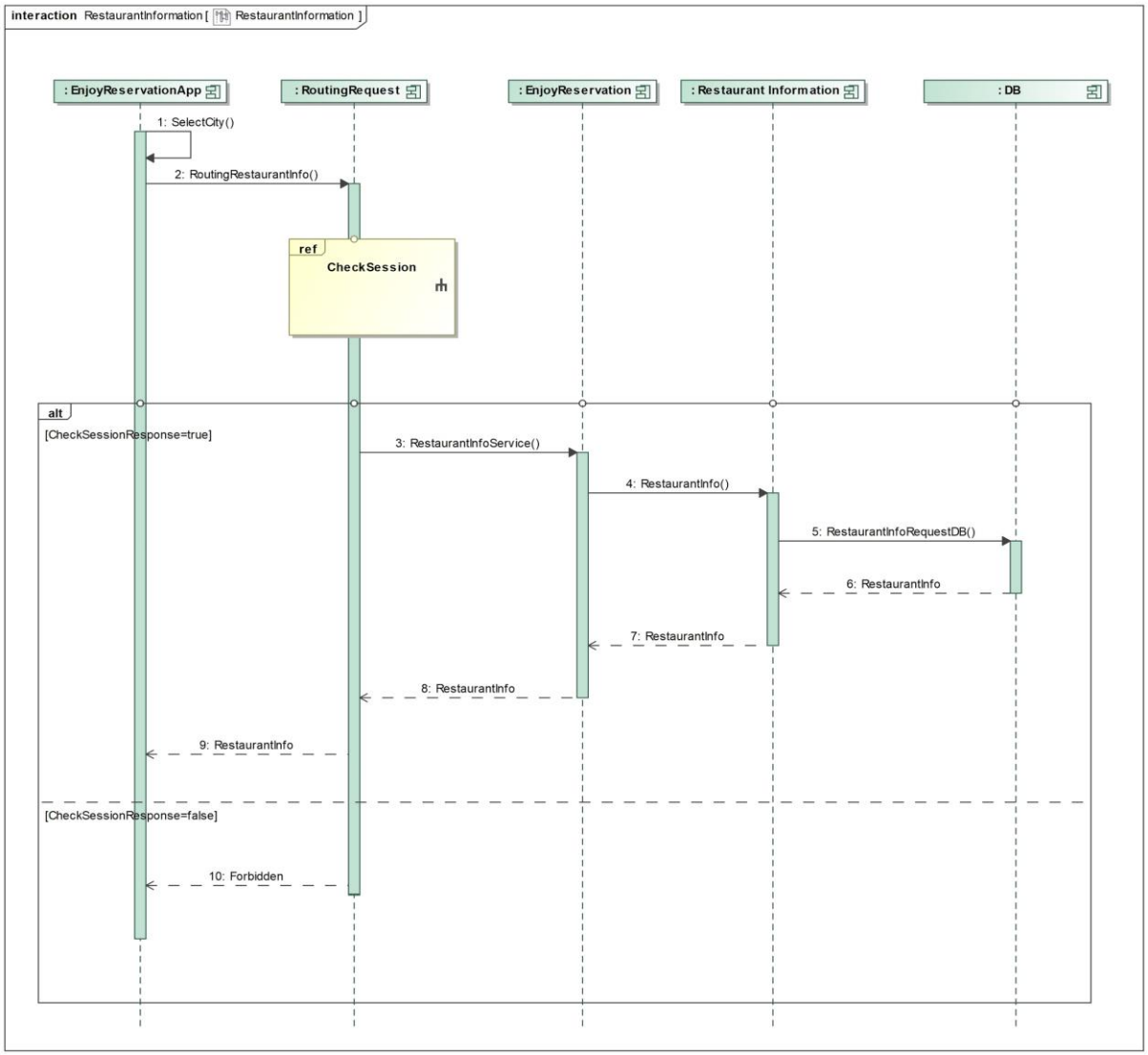

Figure 7. Sequence diagram of the Restaurant Information service in the Enjoy Reservation system

RESTAURANT SINGLE INFORMATION

The figure 8 which is given below describes the Restaurant Single Information service in the Enjoy Reservation system. The user selects the restaurant that he wants to see the information. EnjoyReservationApp sends the restaurant_id by a REST API via a GET request. First of all, the session is checked. If the user is logged, the request will be forwarded to the the EnjoyReservation component and subsequently to the RestaurantInformation component that will query the Database in order to get all the information about the restaurant. A response containing all the information of that restaurant is sent to EnjoyReservationApp.

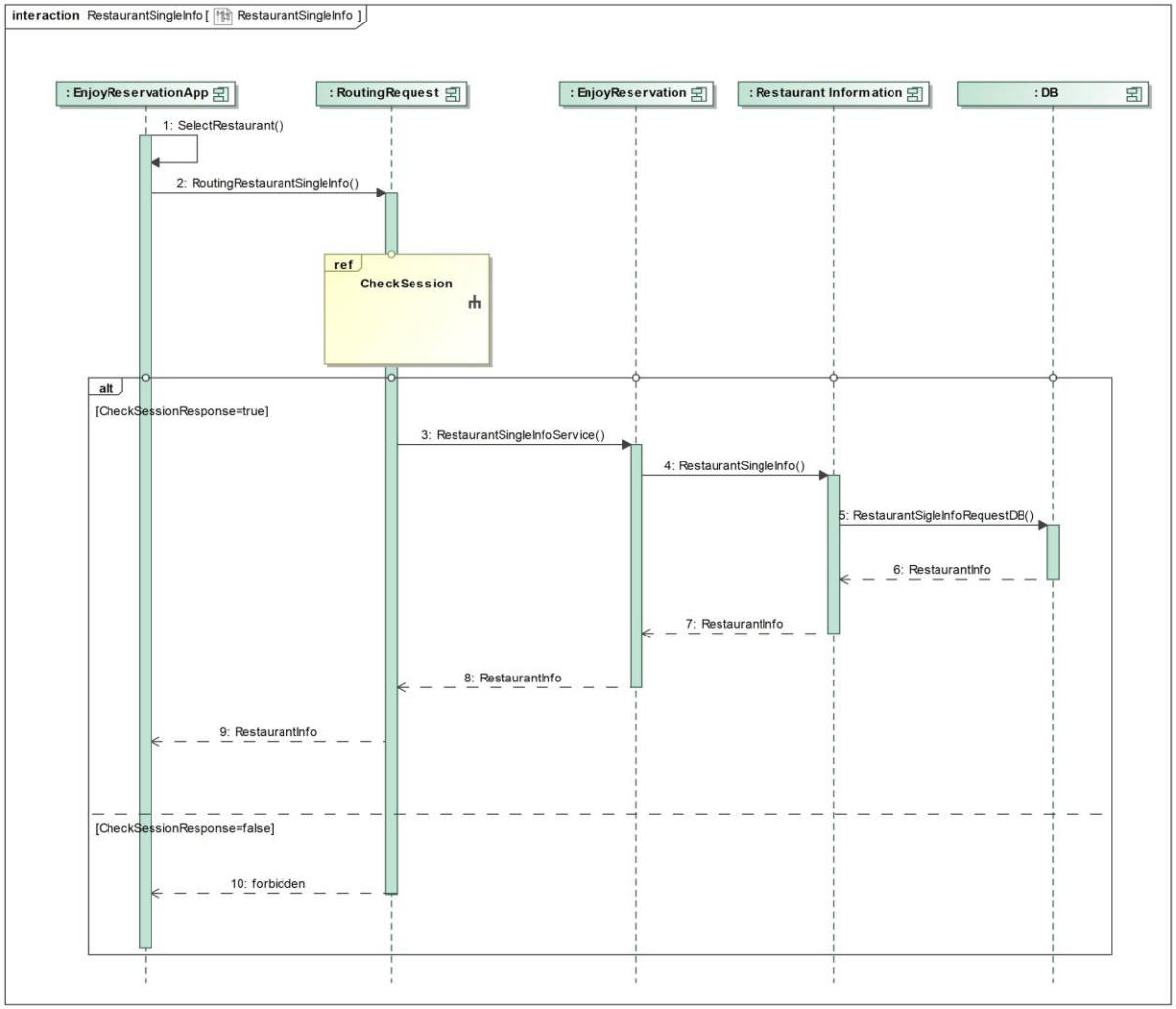

Figure 8. Sequence diagram of the Restaurant Single Information service in the Enjoy Reservation system

The figure 9 which is given below describes the Restaurant Update service in the Enjoy Reservation system. The user selects the restaurant that he wants to update. First of all, the session is checked. If the user is logged, he will receive the information about that restaurant (ref 'Restaurant Single Information'). At this point the user update the restaurant from EnjoyReservationApp and this latter sends this restaurant by a REST API via a POST request. If the user is the restaurant's owner, the request will be forwarded to the the EnjoyReservation component and subsequently to the RestaurantInserting component that will query the Database in order to update the restaurant. A response containing a boolean answer is sent to EnjoyReservationApp.

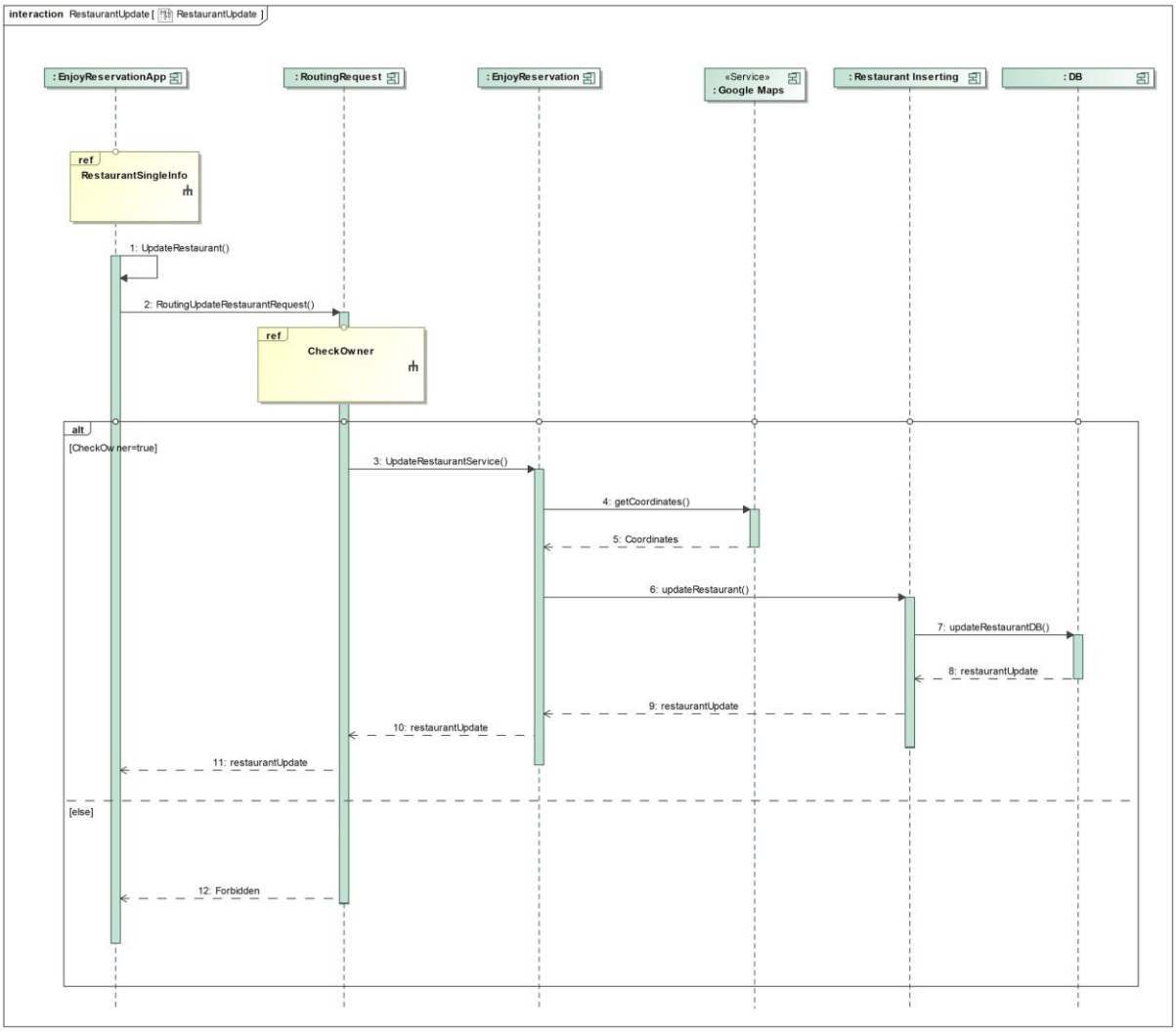

Figure 9. Sequence diagram of the Restaurant Update service in the Enjoy Reservation system

The figure 10 which is given below describes the Restaurant Delete service in the Enjoy Reservation system. The user selects the restaurant that he wants to delete. First of all, the session is checked. If the user is logged, he can delete the restaurant from EnjoyReservationApp and this latter sends this restaurant by a REST API. If the user is the restaurant's owner, the request will be forwarded to the the EnjoyReservation component and subsequently to the RestaurantInserting component that will query the Database in order to delete the restaurant. A response containing a boolean answer is sent to EnjoyReservationApp.

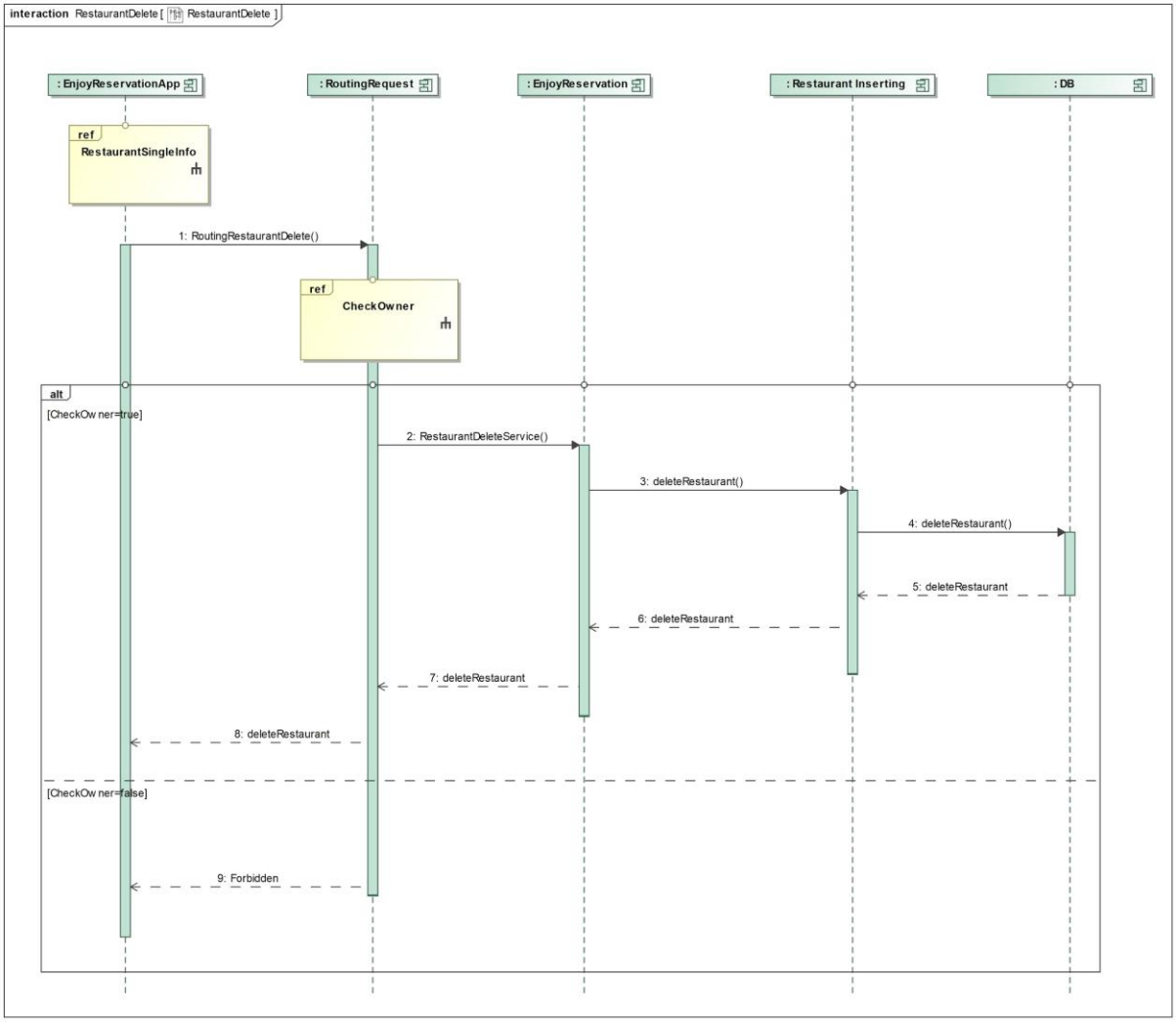

Figure 10. Sequence diagram of the Restaurant Delete service in the Enjoy Reservation system

The figure 11 which is given below describes the Restaurant Booking service in the Enjoy Reservation system. The user selects the restaurant that he wants to book. First of all, the session is checked. If the user is logged, he can book the restaurant from EnjoyReservationApp and this latter sends the booking information by a REST API via a POST request. The request will be forwarded to the the EnjoyReservation component and subsequently to the RestaurantBooking component that will query the Database in order to insert the booking information. A response containing a boolean answer is sent to EnjoyReservationApp.

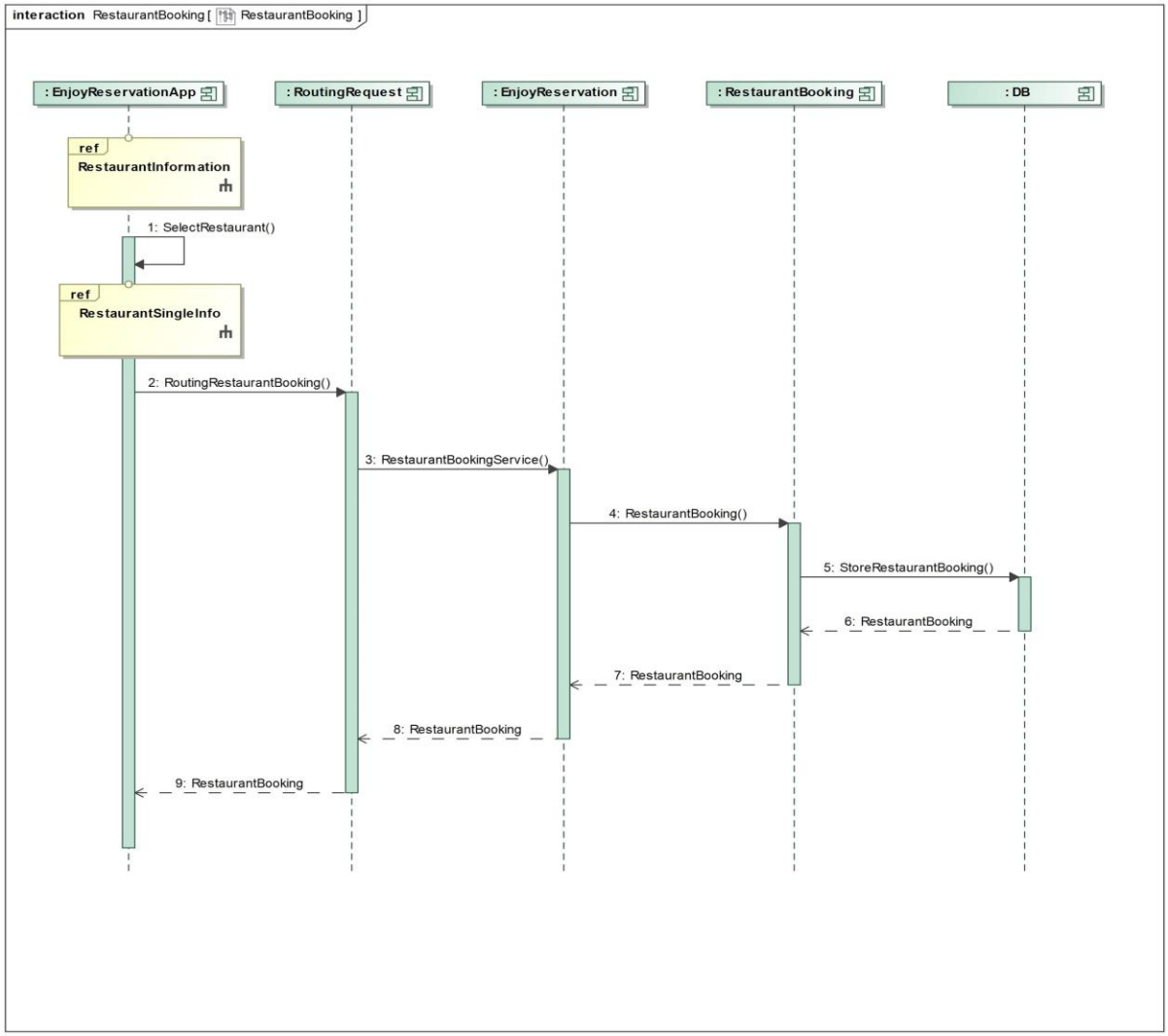

Figure 11. Sequence diagram of the Restaurant Booking service in the Enjoy Reservation system

The figure 12 which is given below describes the Cinema Inserting service in the Enjoy Reservation system. EnjoyReservationApp sends a JSON containing the information about the cinema that the owner wants to roll in the system to the RoutingRequest by a REST API via a POST request. First of all, the session is checked. If the user is logged, the request will be forwarded to the the EnjoyReservation component and subsequently to the CinemaInserting component that will deal with to insert the cinema on the Database. A response containing a boolean answer is sent to EnjoyReservationApp.

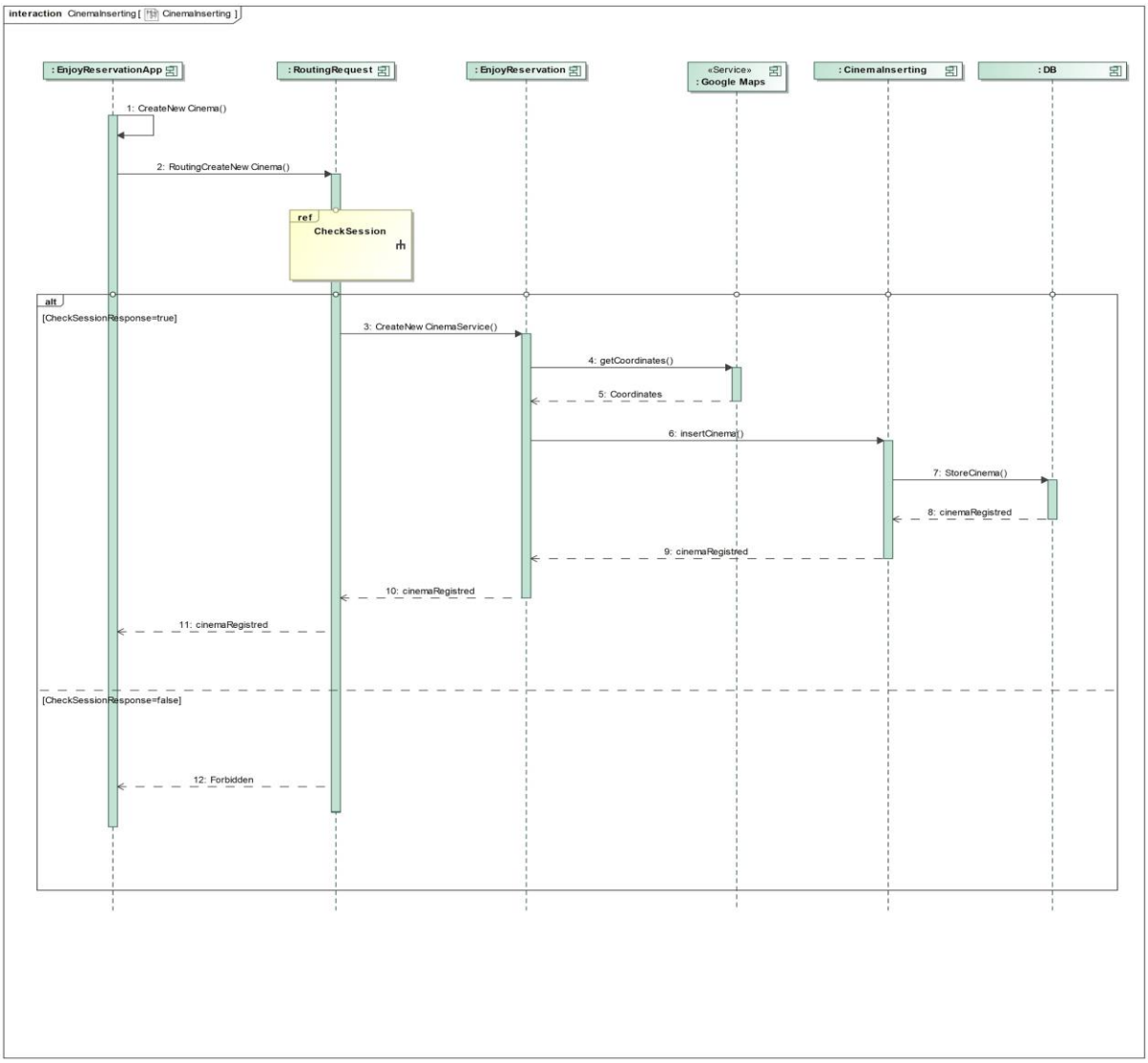

Figure 12. Sequence diagram of the Cinema Inserting service in the Enjoy Reservation system

CINEMA INFORMATION

The figure 13 which is given below describes the Cinema Information service in the Enjoy Reservation system. The user selects the city where he wants to see the cinemas information. EnjoyReservationApp sends the city by a REST API via a GET request. First of all, the session is checked. If the user is logged, the request will be forwarded to the EnjoyReservation component and subsequently to the CinemaInformation component that will query the Database in order to get all the cinemas in a precise city. A response containing a list of cinemas is sent to EnjoyReservationApp.

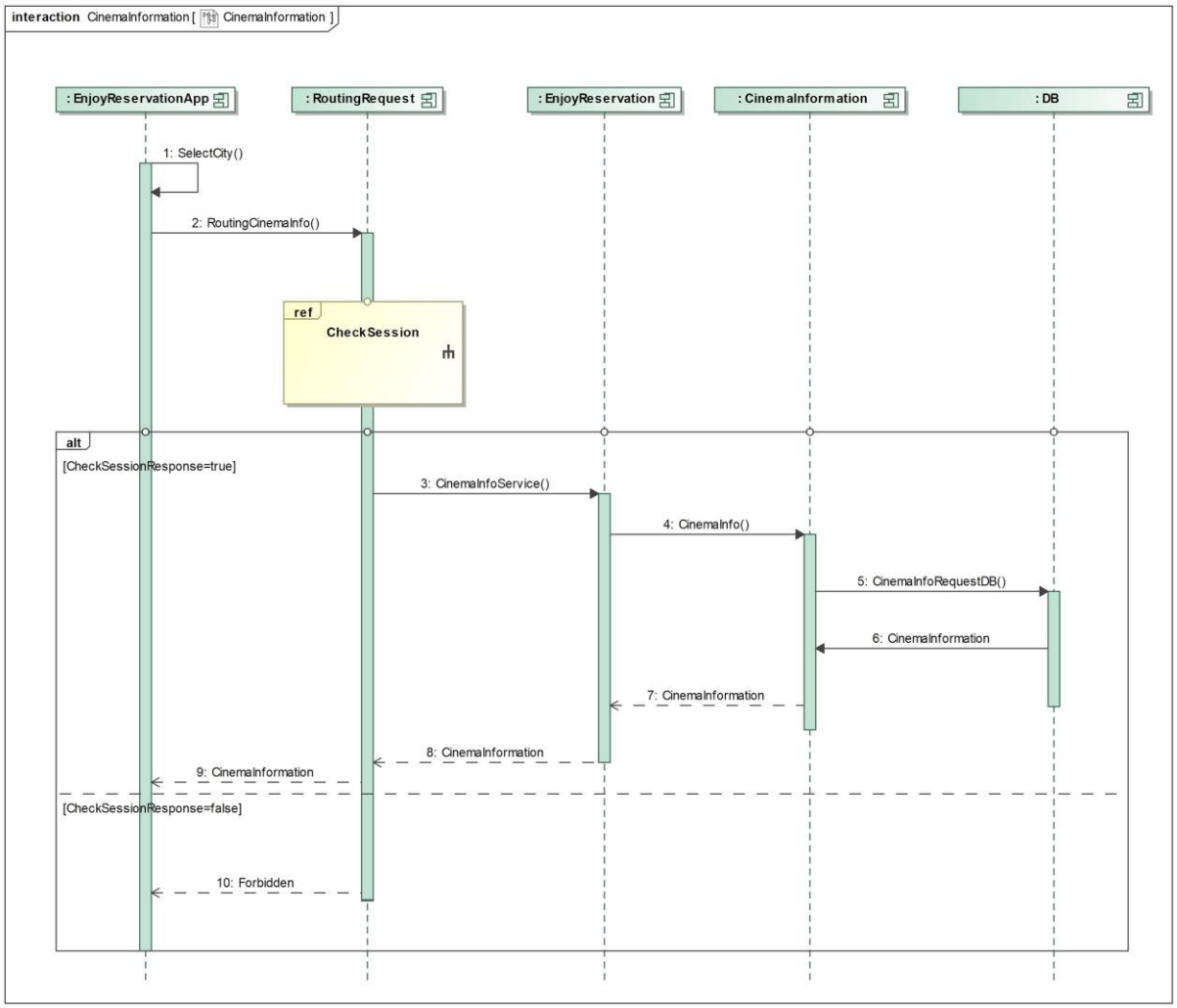

Figure 13. Sequence diagram of the Cinema Information service in the Enjoy Reservation system

CINEMA SINGLE INFORMATION

The figure 14 which is given below describes the Cinema Single Information service in the Enjoy Reservation system. The user selects the cinema that he wants to see the information. EnjoyReservationApp sends the cinema_id by a REST API via a GET request. First of all, the session is checked. If the user is logged, the request will be forwarded to the the EnjoyReservation component and subsequently to the CinemaInformation component that will query the Database in order to get all the information about the cinema. A response containing all the information of that cinema is sent to EnjoyReservationApp.

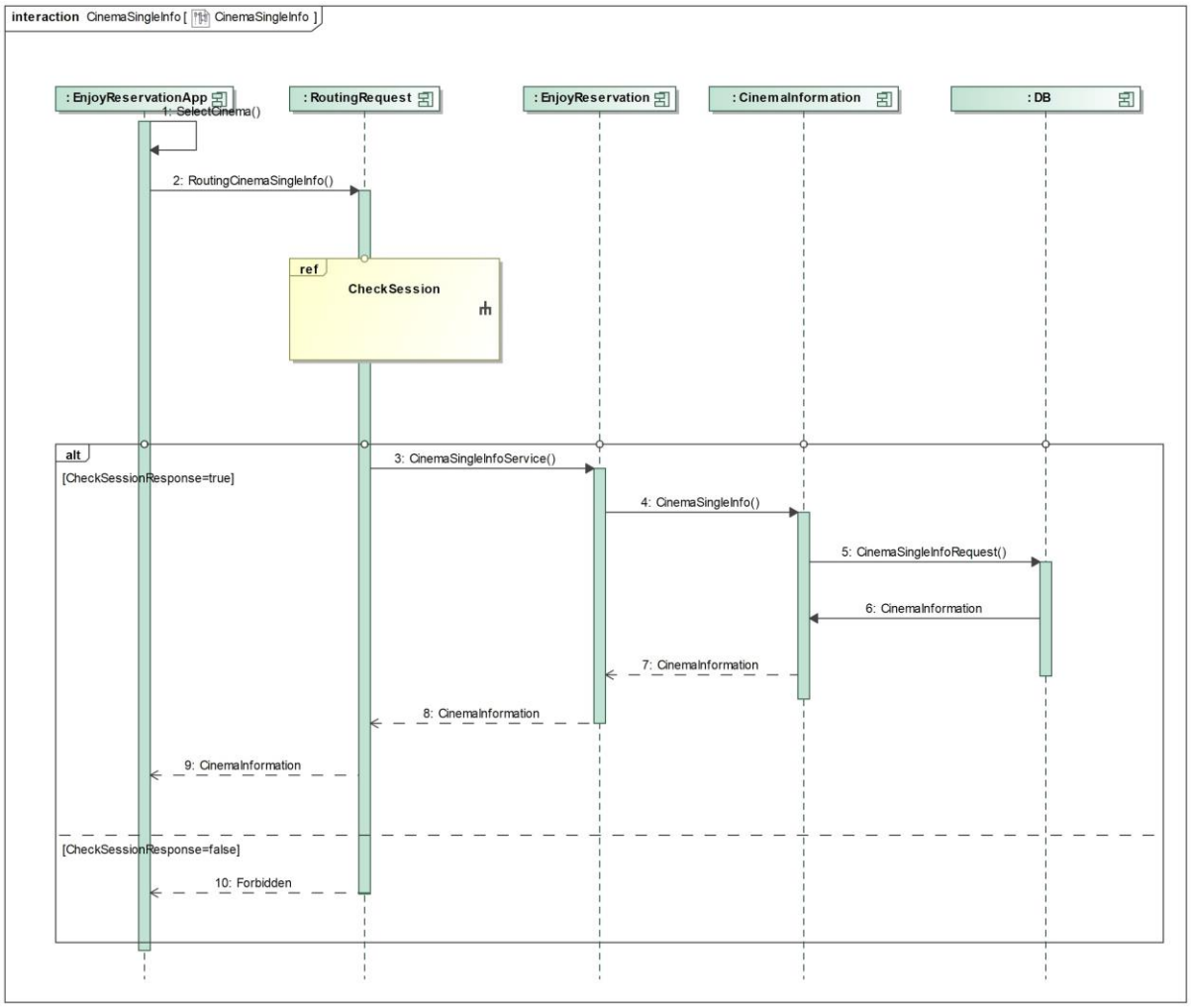

Figure 14. Sequence diagram of the Cinema Single Information service in the Enjoy Reservation system

CINEMA UPDATE

The figure 15 which is given below describes the Cinema Update service in the Enjoy Reservation system. The user selects the cinema that he wants to update. First of all, the session is checked. If the user is logged, he will receive the information about that cinema (ref 'Cinema Single Information'). At this point the user update the cinema from EnjoyReservationApp and this latter sends this cinema by a REST API via a POST request. If the user is the restaurant's owner, the request will be forwarded to the the EnjoyReservation component and subsequently to the CinemaInserting component that will query the Database in order to update the cinema. A response containing a boolean answer is sent to EnjoyReservationApp.

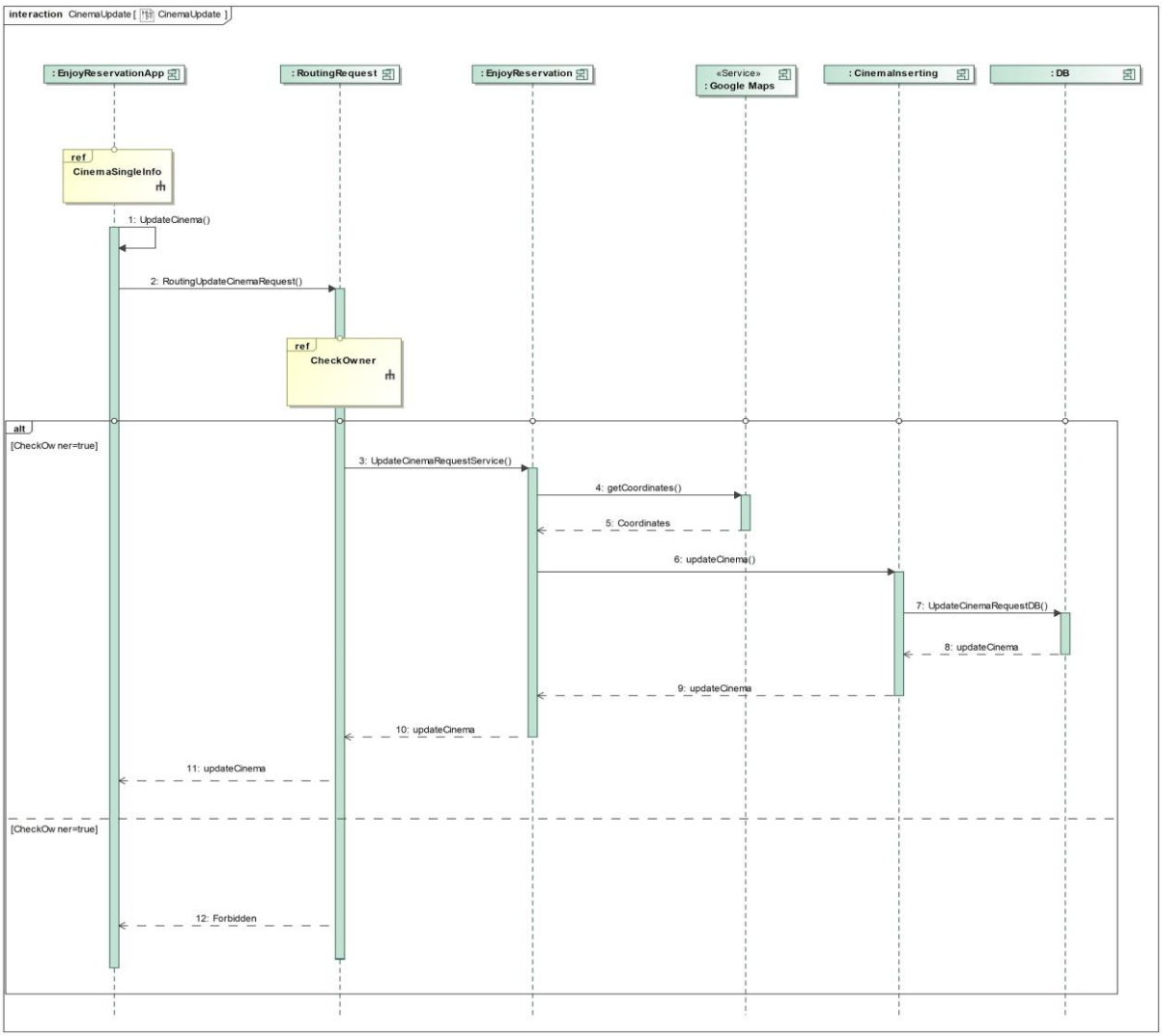

Figure 15. Sequence diagram of the Cinema Update service in the Enjoy Reservation system

CINEMA DELETE

The figure 16 which is given below describes the Cinema Delete service in the Enjoy Reservation system. The user selects the cinema that he wants to delete. First of all, the session is checked. If the user is logged, he can delete the cinema from EnjoyReservationApp and this latter sends this cinema by a REST API. If the user is the restaurant's owner, the request will be forwarded to the the EnjoyReservation component and subsequently to the CinematInserting component that will query the Database in order to delete the cinema. A response containing a boolean answer is sent to EnjoyReservationApp.

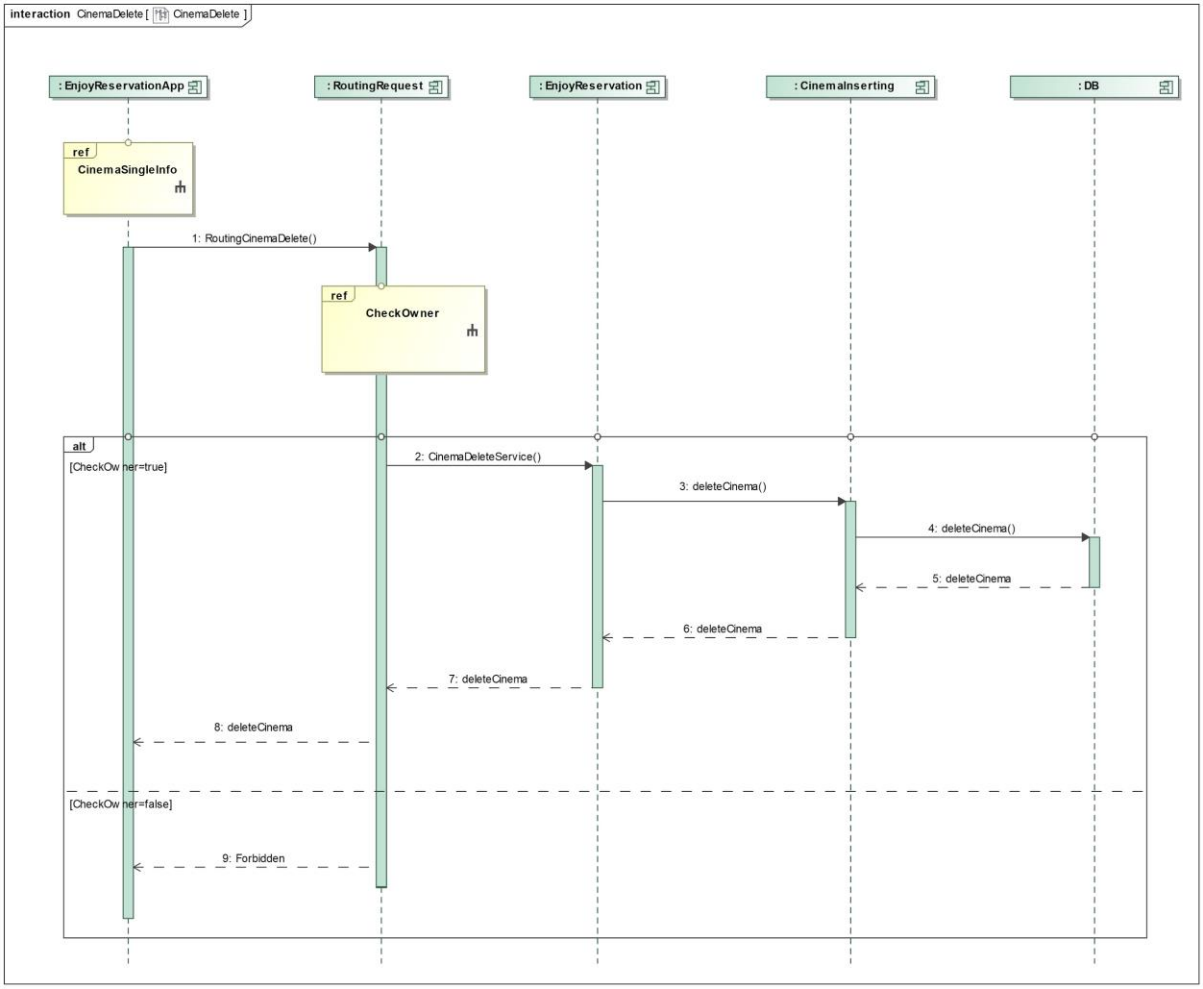

Figure 16. Sequence diagram of the Cinema Delete service in the Enjoy Reservation system

The figure 17 which is given below describes the Cinema Booking service in the Enjoy Reservation system. The user selects the cinema that he wants to book. First of all, the session is checked. If the user is logged, he can book the cinema from EnjoyReservationApp and this latter sends the booking information by a REST API via a POST request. The request will be forwarded to the the EnjoyReservation component and subsequently to the CinemaBooking component that will query the Database in order to insert the booking information. A response containing a boolean answer is sent to EnjoyReservationApp.

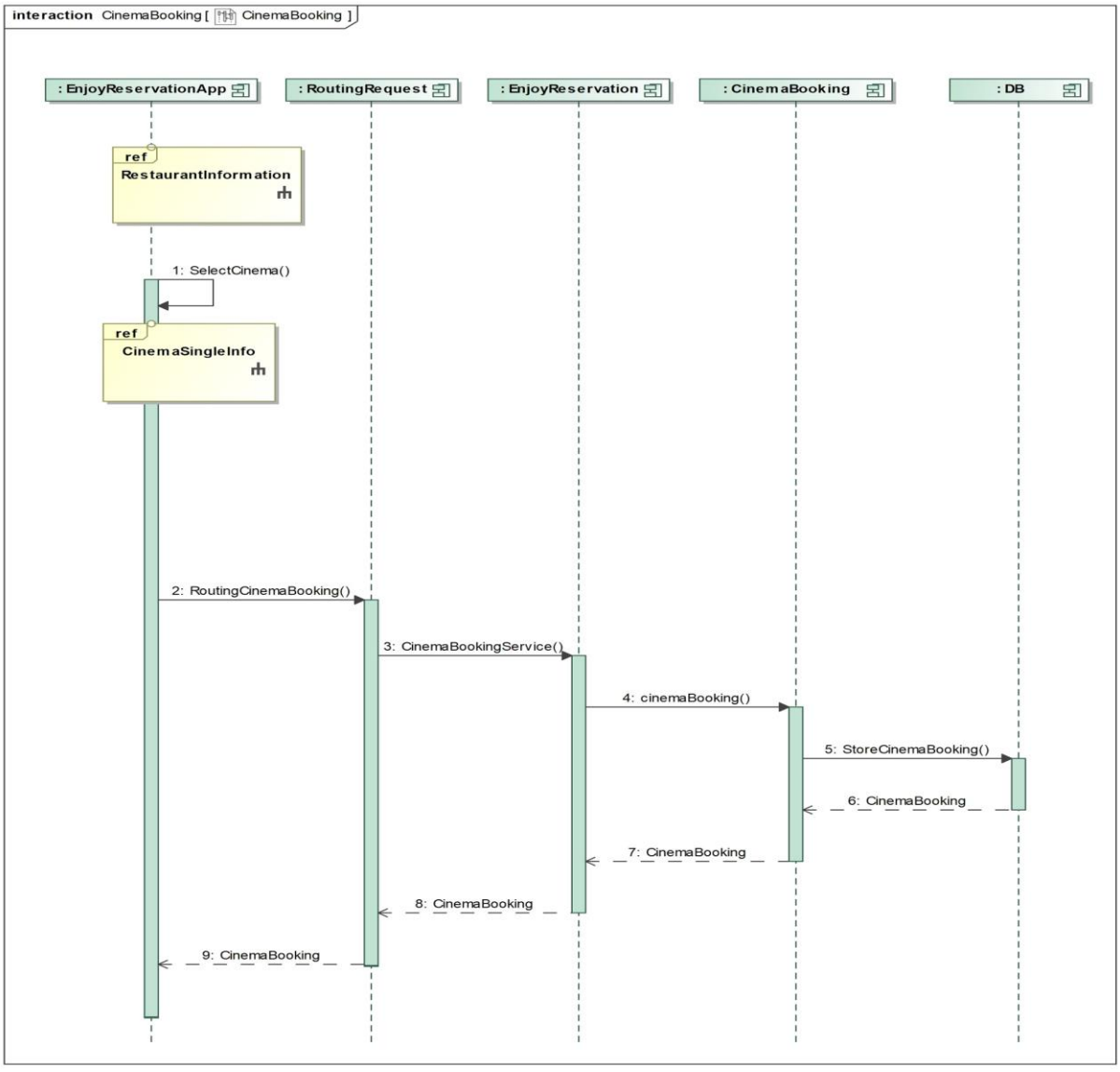

Figure 17. Sequence diagram of the Cinema Booking service in the Enjoy Reservation system

The figure 18 which is given below describes the Night Information service in the Enjoy Reservation system. The user selects the city where he wants to see the restaurants and cinemas information. EnjoyReservationApp sends the city by a REST API via a GET request. First of all, the session is checked. If the user is logged, the request will be forwarded to the the EnjoyReservation component and subsequently to the RestaurantIinformation and the CinemaInformation component that will query the Database in order to get all the restaurants and the cinemas in a precise city. A response containing a list of restaurants and a list of cinemas is sent to EnjoyReservationApp.

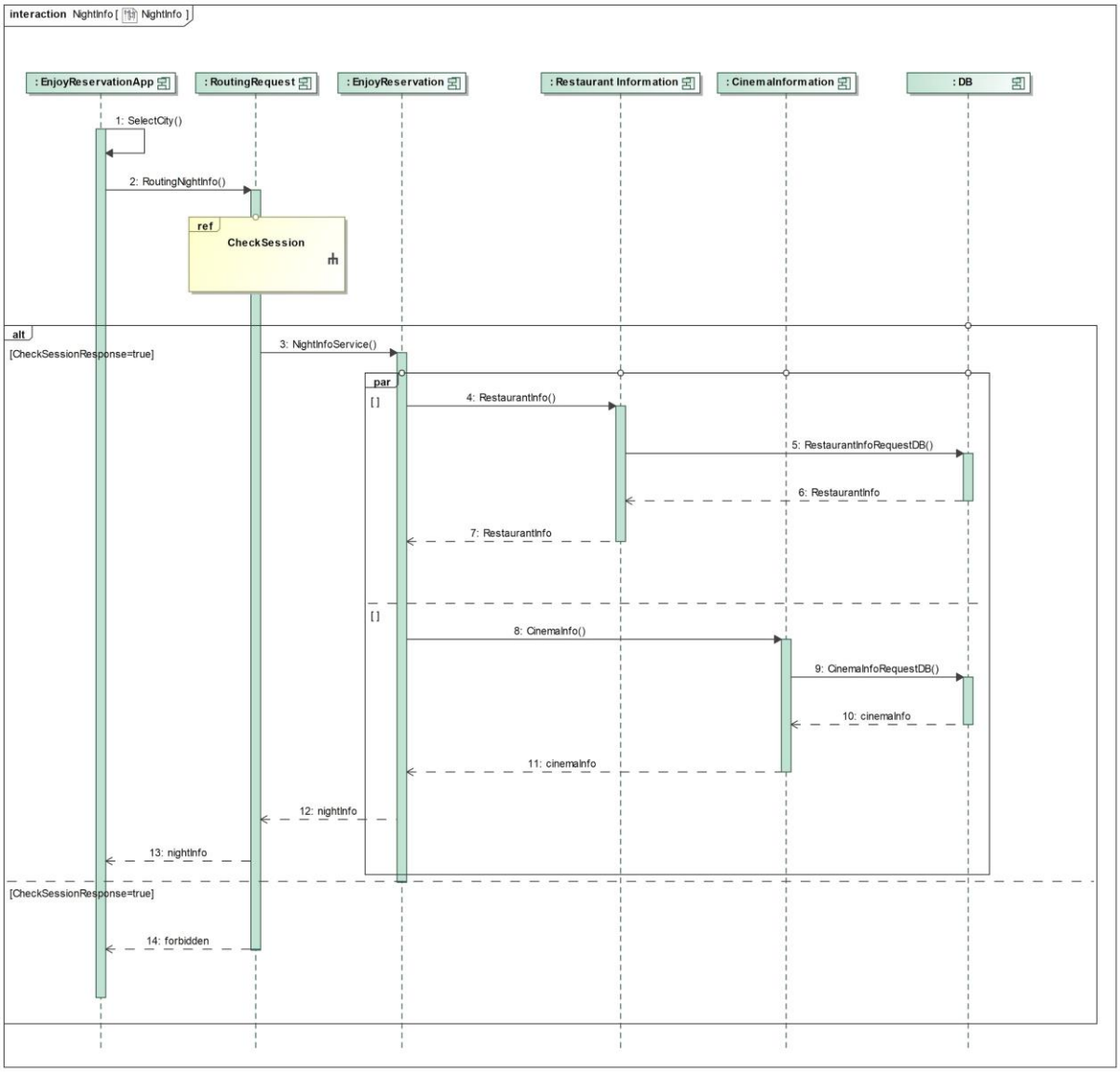

Figure 18. Sequence diagram of the Night Information service in the Enjoy Reservation system

INSTALLATION

In this section the technologies and tools used are listed and a brief description of how install and try out Enjoy Reservation system is explained.

TECHNOLOGIES AND TOOLS

Following table shows the technologies and tools used in Enjoy Reservation system.

Tools and Technology	Version	Download URL
JDK	1.8	http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html
Eclipse Oxygen	2	http://www.eclipse.org/downloads/eclipse-packages/
Tomcat	8.5	https://tomcat.apache.org/download-80.cgi
Maven	3.5.2	https://maven.apache.org/download.cgi#

Download and install the tools and set the environment variables as given in the installation document of respective tools. Below are the few setups we need for this application.

- Download the Github Repository at the following link: https://github.com/CristCapo/SoSe.
- Open all the Maven applications in Eclipse Oxygen, except for the “routingrequests” one.
- Please refer "cald.sql" file inside the folder database in each project to create Tables in MySQL Database, which are required by all the applications.
- Running all the applications that you have just opened on Tomcat within Eclipse. Probably you need to increase Tomcat timeout because the server may take time to run all the applications. For this purpose, please: Open the Servers view -> double click tomcat -> drop down the Timeouts section.
  There you can increase the startup time for each particular server. We suggest to set it at one minute.
- You have to add the “enjoRreservation” application in your local repository. For this purpose, open command prompt, go to enjoyReservation folder and execute following command: *mvn install*.
- You can now build and run the controller application “routingrequests” using following command: *mvn clean package*. Then open command prompt, go to target folder and execute following command: *java -jar routingrequests-1.0.0.jar*.

You can now try out all the services we developed using the REST APIs described in the next section.

API REST

Here you can find a list of API REST built to make the service usable by everyone. Our 'Enjoy Reservation App' implemented for this project uses this API. Note that 'Enjoy Reservation App' is just a test App, it is not complete. If anyone wants in future build a more complete app or a web application to use Enjoy Reservation Service can access the service through this APIs listed below.

ACCOUNT CONTROL

URL: cald/signup

HTTP verb: POST

Header: 'Content-Type: application/json'

Payload:

```
{
   "name": "Luca",
   "surname": "Grillo",
    "email": "lucag.8595@gmail.com",
    "password": "luca",
    "username": "lucgril"
}
```

Takes user data as input as shown above. Returns the session token of the newly registered user

URL: cald/login

HTTP verb: POST

Header: 'Content-Type: application/json'

Payload:

```
{
    "email": "lucag.8595@gmail.com",
    "password": "luca"
}
```

This API takes an object containing a user's username and password as input and returns an object containing its session token.

URL: cald/logout/{token}

HTTP verb: DELETE

Header: -

Payload: -

Delete the session token in the url from the database.

CINEMA CONTROLLER

URL: cald/cinema/{token}/information/city/{city}

HTTP verb: GET

Header: -

Payload: -

This API returns the information of all the cinemas of the city passed in input in the url.

URL: cald/cinema/{token}/information/{id}

HTTP verb: GET

Header: -

Payload: -

This API takes returns the information of the cinema specified in the “id” Parameter.

URL: cald/cinema/{token}/booking

HTTP verb: POST

Header: 'Content-Type: application/json'

Payload:

```
{
  "id_hall": 1,
  "id_film": 1,
  "id_utente": 1,
  "seats": 2,
  "schedule": "2018-02-01T22:00:00"
}
```

This API takes an object containing information about hall, film, userID, number of seats to book and time of the film as input and returns an object containing a message.

URL: cald/cinema/{token}/insert

HTTP verb: POST

Header: 'Content-Type: application/json'

Payload:

```
{
    "lat": 0,
    "lon": 0,
    "name": "Prova",
    "address": "Via Giuseppe Mazzini",
    "cap": "37121",
    "city": "Verona",
    "telephoneNumber": "0863241515",
    "hall": [
                {
                      "number": 1,
                      "seatsNumber": 80,
                       "hallInfo": [
                                        {
                                         "time": "2018-02-18T16:00:00",
                                         "film":
                                          {
                                             "name": "Avengers: Infinity War",
                                             "director": "Anthony Russo",
                                             "cast": "Robert Downey Jr, Josh Brolin"
                                             "duration": 147,
                                             "type": "Action",
                                             "plot": "Four years after the events of…..."
                                          },
                                         "price": 5,
                                         "freeSeatsNumber": 248
                                        }
                                    ]
                                },
                {
                  "number": 1,
                  "seatsNumber": 80,
                  "hallInfo":
                            [
                              {
                                    "time": "2018-02-18T20:00:00",
                                     "film":
                                      {
                                           "name": "Avengers: Infinity War 2",
                                           "director": "Anthony Russo",
                                           "cast": "Robert Downey Jr, Josh Brolin",
                                           "duration": 147,
                                           "type": "Action",
                                           "plot": "Four years after the events……..."
                                      },
                                    "price": 7.5,
                                    "freeSeatsNumber": 110
                                }
                            ]
            }
        ]
}
```


This API takes an object containing all the information about a cinema as input and returns an object containing a boolean. Note that you can add one or more halls, within which can be one or more hallInfo (used in order to register the schedule of the cinema).

URL: cald/cinema/{token}/update

HTTP verb: PUT

Header: 'Content-Type: application/json'

Payload:

```
{
    "id": 6,
    "lat": 1,
    "lon": 1,
    "name": "ProvaUpdated",
    "address": "Via Giuseppe Mazzini",
    "cap": "37121",
    "city": "Verona",
    "telephoneNumber": "0863241515",
    "hall": [
                {
                      "hall_id": 5,
                      "number": 1,
                      "seatsNumber": 80,
                       "hallInfo": [
                                        {
                                         "id": 4,
                                         "time": "2018-02-18T16:00:00",
                                         "film":
                                          {  "id": 1,
                                             "name": "Avengers: Infinity War",
                                             "director": "Anthony Russo",
                                             "cast": "Robert Downey Jr, Josh Brolin"
                                             "duration": 147,
                                             "type": "Action",
                                             "plot": "Four years after the events of…..."
                                          },
                                         "price": 5,
                                         "freeSeatsNumber": 248
                                        }
                                    ]
                                },
                {  "number": 1,
                  "seatsNumber": 80,
                  "hallInfo":
                              [
                                {
                                      "id": 5,
                                      "time": "2018-02-18T20:00:00",
                                       "film":
                                        {
                                             "id": 1,
                                             "name": "Avengers: Infinity War 2",
                                             "director": "Anthony Russo",
                                             "cast": "Robert Downey Jr, Josh Brolin",
                                             "duration": 147,
                                             "type": "Action",
                                             "plot": "Four years after the events…….."
                                          },
                                      "price": 7.5,
                                      "freeSeatsNumber": 110
                                }
                              ]
            }
        ] }
```

This API takes an object containing all the information about a cinema as input and returns an object containing a boolean. Once checked if the user is the owner of the cinema, from this API the user can modify a cinema previously stored in the DB. Through this API the user can also modify halls, films and schedule.

URL: cald/cinema/{token}/delete/{id}

HTTP verb: DELETE

Header: -

Payload: -

This API takes the id of the schedule to be deleted and the user token from the url as input. Once checked if the user is the owner of the cinema, he can delete the schedule of one film from the DB. It is useful when the owner of a cinema needs to delete some films from the program.

RESTAURANT CONTROLLER

URL: cald/restaurant/{token}/information/city/{city}

HTTP verb: GET

Header: -

Payload: -

This API takes returns the information of all the restaurants of the city passed in input in the url.

URL: cald/restaurant/{token}/information/{id}

HTTP verb: GET

Header: -

Payload: -

This API takes returns the information of the restaurant specified in the “id” Parameter.

URL: cald/restaurant/{token}/booking

HTTP verb: POST

Header: 'Content-Type: application/json'

Payload:

```
{
  "restaurant": 1,
   "user": 1,
   "seats": 2,
   "schedule": "2018-02-10T20:00:00"
}
```

This API takes an object containing information about restaurant, userID, number of seats to book and time of the lunch/dinner as input and returns an object containing a message.

URL: cald/restaurant/{token}/insert

HTTP verb: POST

Header: 'Content-Type: application/json'

Payload:

```
{
    "lat": 0,
    "lon": 0,
    "name": "CiaoCiao",
    "address": "Via Monte Brancastello 2",
    "cap": "67100",
    "city": "L'Aquila",
    "telephoneNumber": "1234",
    "style": "bo",
    "cuisine": "vo",
    "menu": "no",
    "maxSeats": 90,
    "discount": {
    "cinema": 3,
    "price": 12
}
```

This API takes an object containing all the information about a restaurant and a related discount as input and returns an object containing a boolean.

URL: cald/restaurant/{token}/update

HTTP verb: PUT

Header: 'Content-Type: application/json'

Payload:

```
{
    "id": 6,
    "lat": 1,
    "lon": 1,
    "name": "CiaoCiaoUpdated",
    "address": "Via Monte Brancastello 2",
    "cap": "67100",
    "city": "L'Aquila",
    "telephoneNumber": "1234",
    "style": "Italian",
    "cuisine": "Tipical",
    "menu": "no",
    "maxSeats": 90,
    "discount": {
    "id": 5,
    "cinema": 3,
    "price": 12
}
```

This API takes an object containing all the information about a restaurant and his discount as input and returns an object containing a boolean. Once checked if the user is the owner of the cinema, from this API the user can modify a restaurant previously stored in the DB. Through this API the user can also modify discount.

URL: cald/restaurant/{token}/delete/{id}

HTTP verb: DELETE

Header: -

Payload: -

This API takes takes the id of the restaurant to be deleted and the user token from the url as input. Once checked if the user is the owner of the restaurant, he can delete the restaurant and the related discount from the DB. It is useful when the owner of a restaurant needs to delete the restaurant from the EnjoyReservation service.

NIGHT CONTROLLER

URL: cald/night/{token}/information/city/{city}

HTTP verb: GET

Header: -

Payload: -

From this API you can send a parallel request of cinema information and restaurant information for a determinate city. This API will return a list of restaurants and a list of cinemas of a determinate city.

We made a very simple mobile application build with Ionic in order to show some functionalities of our services. You can download the Github repository at the following link: https://github.com/CristCapo/SoSeClient.

We made a simple Login page (the figure on the left) and the possibility to see either the Restaurants and the Cinemas of a city (chosen by the user) with a SINGLE REST CALL. You can see the list in the figure on the right. By clicking on a blue button, you can show the information about a single restaurant and you can set a booking. Besides, you can insert, delete or update a restaurant.