M.Sc. in Computer Science and Engineering

Software Engineering 2

Design Document
(DD)

Version 1.4
Released on: 10th December 2016

Authors:

Marilena Coluccia
Burcu Cesur
Mustafa Çeçe

Reference Professor: Elisabetta Di Nitto

INDEX[1]

[1] Clickable index: each title is a link to the correspondent paragraph

TABLE OF CONTENT

1 INTRODUCTION

1.1 Purpose

The aim of this document is to express in further detail what previously described in the RASD of Power EnJoy system.

This document is addressed to developers to identify the architectures and the design of the system, explaining the architectural decisions and tradeoffs chosen in the design process and its justifications.

1.2 Scope

The architectural design provides implementation of general functionalities based on the presented RASD. Since it is assumed in RASD that the old system already has two user types: administrator and employee which are also exist in the database system, design of architecture is made based on two other users: guest and customer.

The system is in communication with GPS, SMS and with cars and receiving information from their system. While targeting the customer, following functionalities are considered for the system architecture.

PowerEnJoy will manage:

- registering of guest and logging in/out of customer.
- car reservation creation
- car reservation cancellation
- changes on profile data of customers

The system also includes functionalities as options such as money saving and special parking areas.

1.3 Definitions, acronyms, abbreviations

RASD: Requirements Analysis and Specifications Document

DD: Design Document

Customer: A person that, after registering, has an account. Customer register to the system with name, surname, mobile number, credit card number and drive license then an access to the system is provided with a password that by sending an SMS to the mobile number used in registration.

Employee: A person that is registered to the system by the administrator and is responsible of maintaining the system flow successful by helping customers and taking care of cars.

Administrator: The administrator of the system is the person that has a full access to all data of the system and he has the authority to manage the system.

Guest: A possible customer who hasn't logged in or registered yet. Guests can have access to the system functionalities only after making a registration and logging in.

Reservation: It can be made for a single car for at most one hour. Once the customer has reserved a car, the reservation is created immediately and expires after one hour.

Missed reservation: If the customer doesn't pick up the car that he has reserved for an hour, the customer is fined of 1 EUR and the car is marked as available by the system.

Canceled reservation: If the customer wants to cancel the reservation, he can make a cancellation request in an amount of money up to 30 minutes after the reservation has been made. The customer is fines for 0.50 EUR if he cancels the reservation.

Charging: Charging is made with a given amount of money per minute. Charging starts as soon as the engine ignites and ends as soon as the engine has stopped, the car has been parked in a safe area and the driver has got off the car.

Payment: Payment is automatically done from the credit card number defined during the registration after a certain amount of time has passed from the charging stop.

Power grid station: Station that is equipped with a certain amount of power grids in order to charge the cars. It is under the control of the company and it is owned by the company.

GPS: The Global Positioning System is a satellite-based navigation system that transmits radio signals. It is used to provide geolocalization and time information. The GPS has a specific role in the system, as it both receives and provides the coordinates of the cars and customers.

Safe Area: It is the only area where parking is allowed. It is shown in the map by default and it contains the electric cars.

Trip: It starts when the customer gets in the cars and starts the engine.

Shared Trip: It is possible to have other people in the car. In order to have a discount, a customer should travel with at least two other passengers. Shared trip is automatically detected with the sensor system.

Shared Trip Discount: If the system detects the customer has taken at least two passengers inside the car, the system applies a discount of 10% on the current trip.

Battery Friendly Discount: If a car is left with at most the half of the battery, the system applies a discount of 20% on the current trip.

Increment in charge: If a car is left with no more than 20% of the battery still working, or if the car is left 3 km far from the nearest power grid station, then the system charges 30% more on the last trip, in order to compensate the cost required to re-charge the car.

SMS: Short Message Service is a form of text-messaging communication based on phones and mobile phones.

API: Application Programming Interface is used for interacting and communicating with other existing systems.

UX: User Experience Design

BCE: Business Controller Entity

JDBC: Java Database communication

Money Saving: It is an option that customer is able to enable. Enabling this option follows requesting a destination input in order to provide information about the station where to leave the car to get a discount. This station is determined to ensure a uniform distribution of cars in the city and depends both on the destination of the user and on the availability of power plugs at the selected station. This stations are already defined in the old system.

Special Parking Areas: These are for to left the cars at so customers can be recharge, take care of plugging the car into the power grid so the system applies a discount of 30% on the last ride. They are already defined in the old system

1.4 Reference documents

The following references were used to specify this document:

- Assignment2016/2017.pdf
- DesignPartI.pdf
- DesignPartII.pdf

1.5 Document structure

- Introduction

This section describes the purpose of the project and its general functional restrictions. Also introduces to the reader the set of important words and language used in this document and the general description of its structure.

- Architectural design

Software application architecture is the process of defining a structured solution that meets all of the technical and operational requirements, while optimizing common quality attributes such as performance, security, and manageability. It involves a series of decisions based on a wide range of factors, and each of these decisions can have considerable impact on the quality, performance, maintainability, and overall success of the application.

- Algorithm design

This section includes the most critical algorithms and how they would be implemented taking into account that part of the methods derived from external architectures.

- User interface design

This section presents mockups and user experience explained via UX and BCE diagrams.

- Requirements traceability

This section aims to explain how the decisions taken in the RASD are linked to design elements.

2 ARCHITECTURAL DESIGN

2.1 Overview

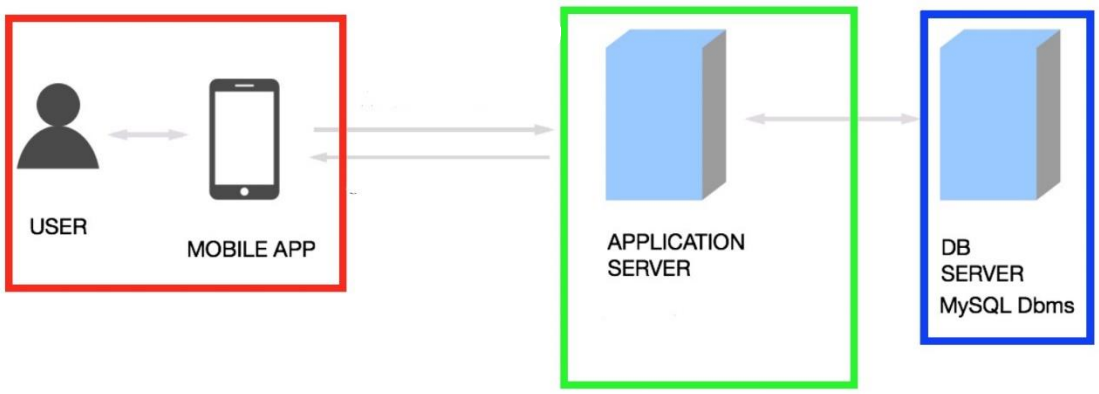

Fig. 2.1 General architecture

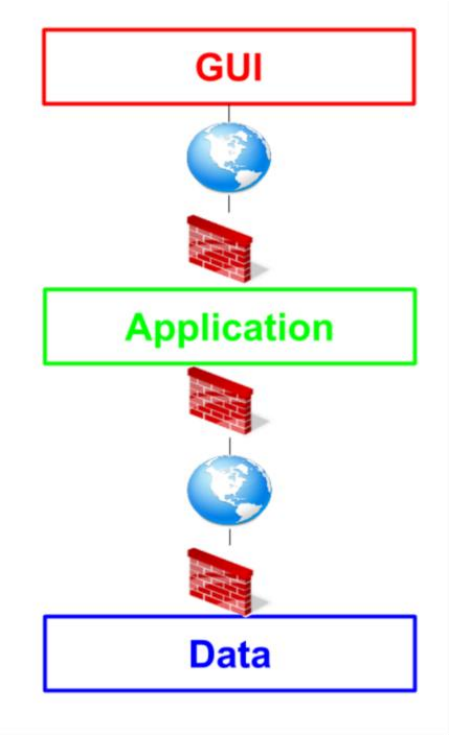

Fig. 2.2 Tiers

The Power EnJoy system was structured in three tiers. It was thought to allow access to the client side only through mobile. In particular, it proposes a dynamic user interface.

The GUI has been designed for mobile but, the fact that all desktop operating systems now support the installation of apps available on the store, it gives the option to have the client side even on personal computers without changing the architecture presented in this document

This structure easily enables the transfer of the system on a cloud platform, as Azure. This is possible since the implementation time, thanks to virtual machines on the Azure platform, that allow you to simulate the pre-production environment, and then set it into production.

2.2 High level components and their interaction

The high level components architecture is composed of four different elements types:

- Controller;
- Customer;
- Car;
- Old system.

The customer initiate the communication from the mobile application.

The customer and the controller exchange information synchronously, except for the completion of the registration to the system, which takes place via SMS.

The car send information/notification about its battery level and location to the controller in an asynchronous way and the controller can change the car availability as result of a reservation synchronously.

It points out that will not be implemented in this project, nor how to collect the information by the machine system nor the way it will be received such information from the main system, as it has been assumed that this architecture and software is already present and well operated

A final type of components is also present, the old system, where there are employees and administrator account, safe areas, power grid and city map. The controller store customer information on the Database and retrieves them when needed (i.e. login)

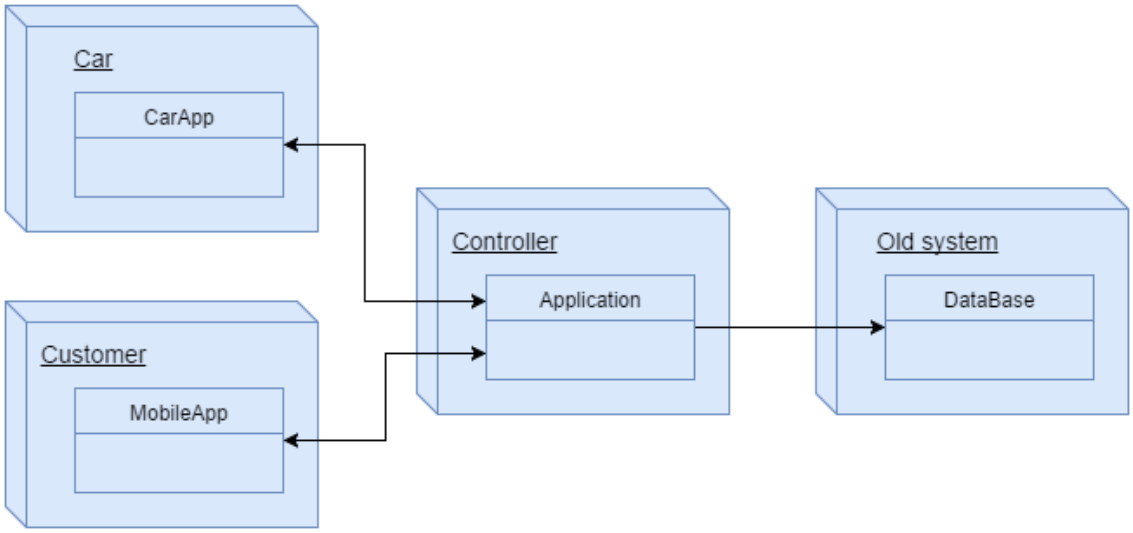

Fig. 2.3 High level components

2.3 Component view

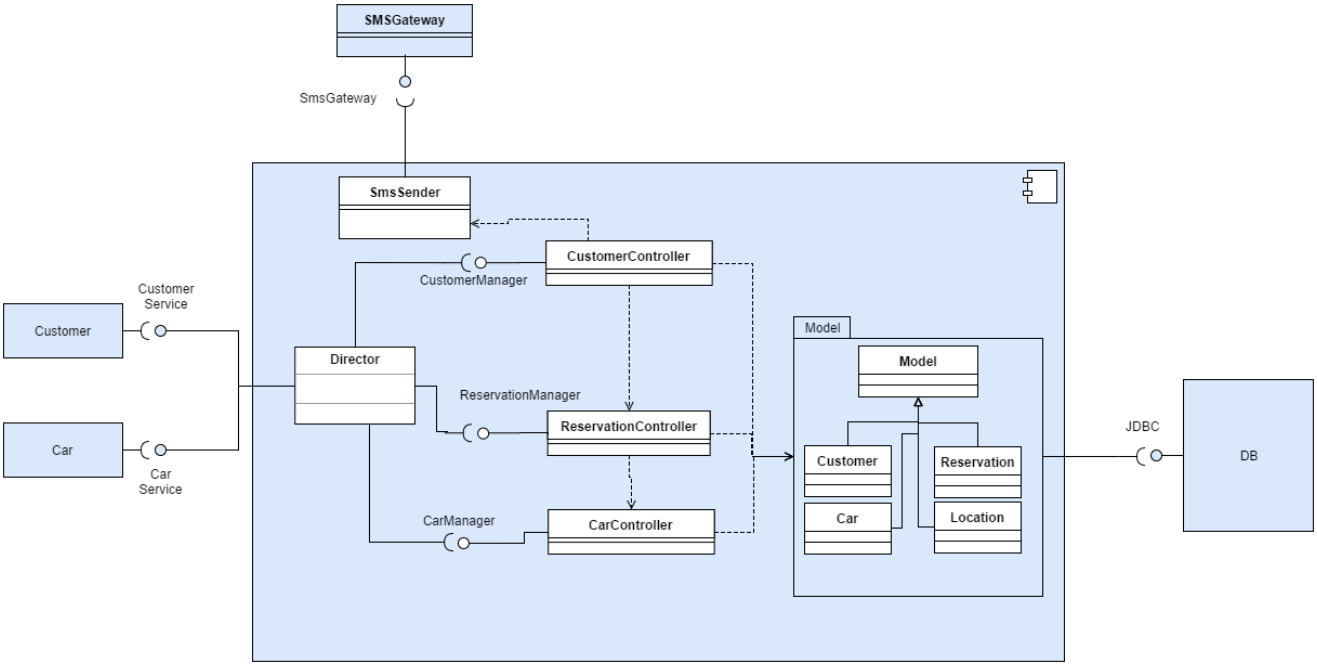

Fig. 2.4 Component view

CAR: the car device.

CUSTOMER: the customer device.

DIRECTOR: it sorts requests received so that they reach the right controller.

SMSSENDER: manage SMS.

CUSTOMERCONTROLLER: manage customer request (i.e. login and account modification).

RESERVATIONCONTROLLER: manage reservation.

CARCONTROLLER: manage car information and utilization.

MODEL: the world modelization, data.

SMSGATEWAY: manage the sending of SMS message.

DB: Database.

2.4 Deploying view

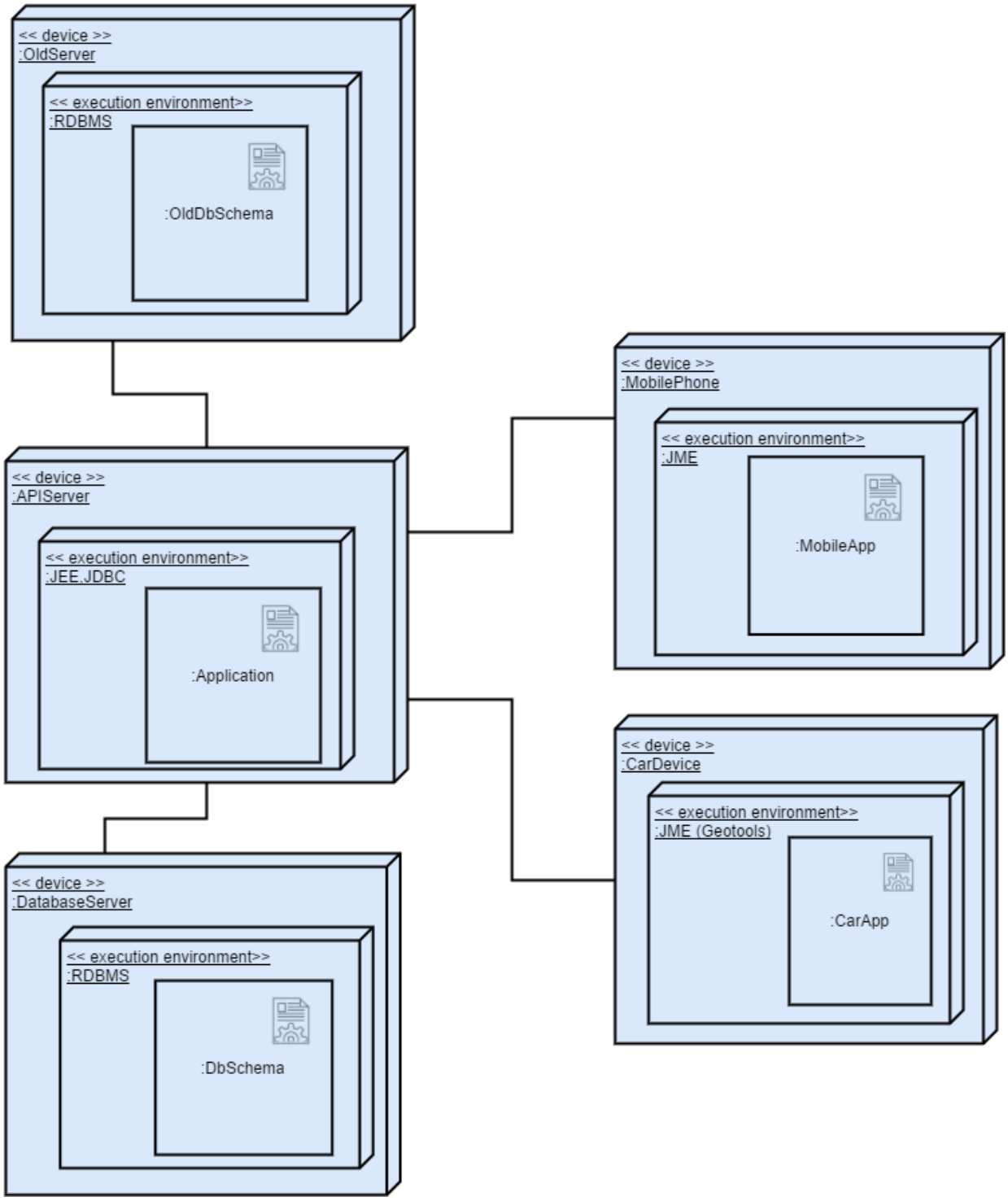

Fig. 2.5 Deployment view

2.5 Runtime view

2.5.1 Registration

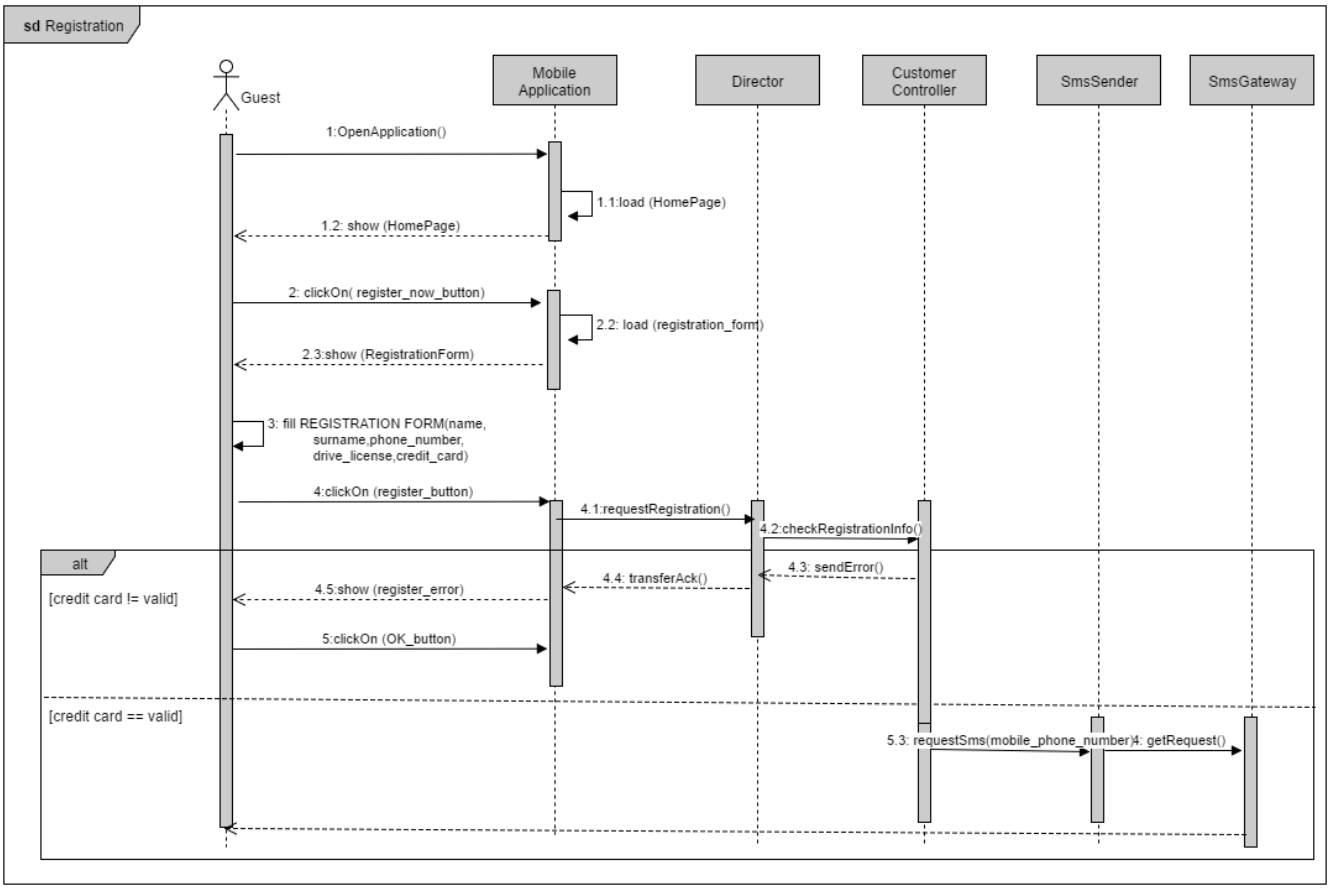

Fig. 2.6 Registration runtime view

To Register to the system (Fig. 2.6) the Guest, after launching the application, needs to click register button and he/she is going to see a registration form and should fill it. After this point, information will be checked by the system. If the process is successful the CustomerController sends the customer a text message, through SmsSender, with the password, otherwise he will notify the error.

2.5.2 Log in

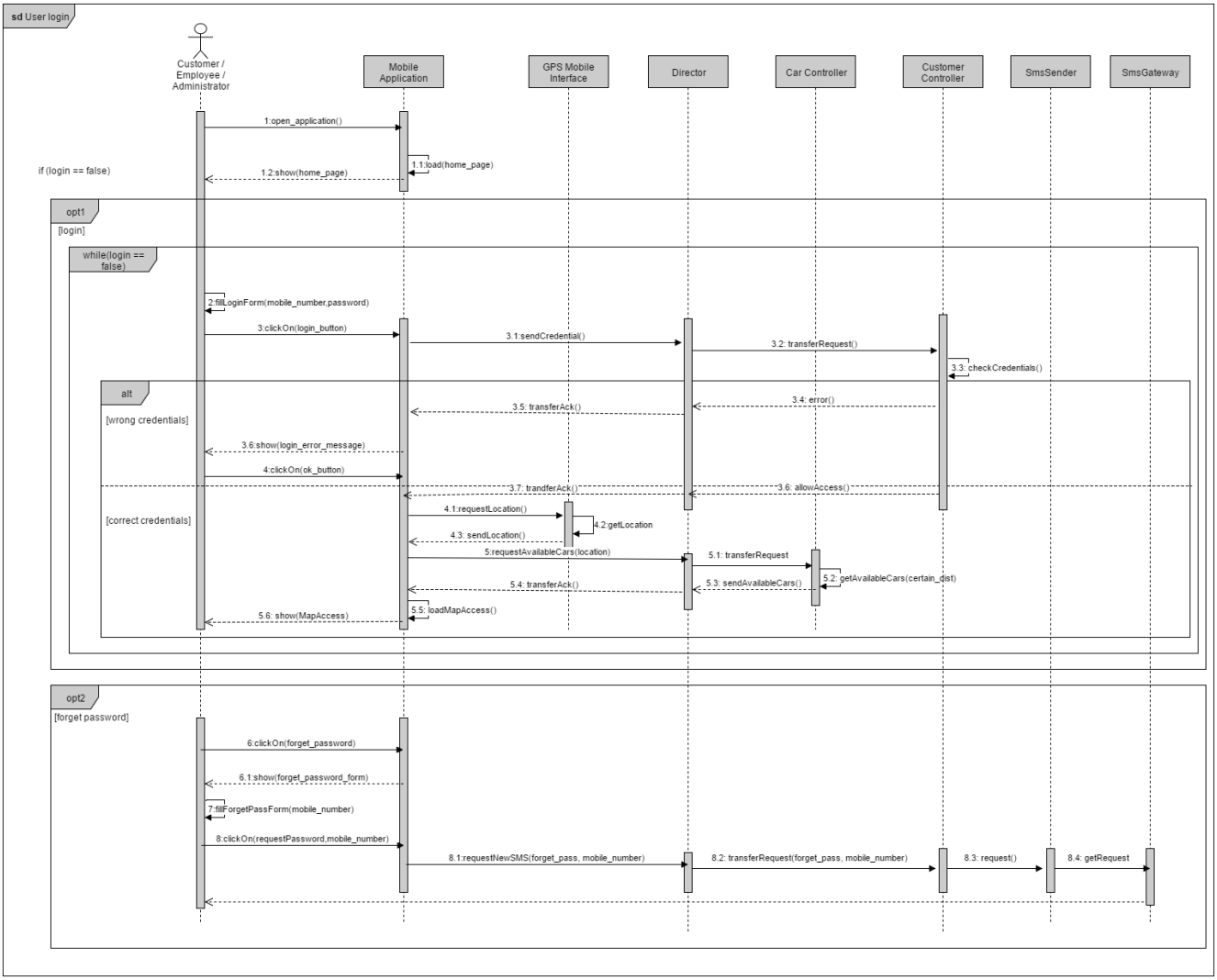

Fig. 2.7 Login runtime view

The diagram in Fig. 2.7 show the Customer Login. There are two option which are opt1 and opt2. In opt1 there will be credential procedure using Customer Controller, Director and GPS. In opt2 the system allows the customer to reset his password, if he lost it sending a new password.

2.5.3 Logout

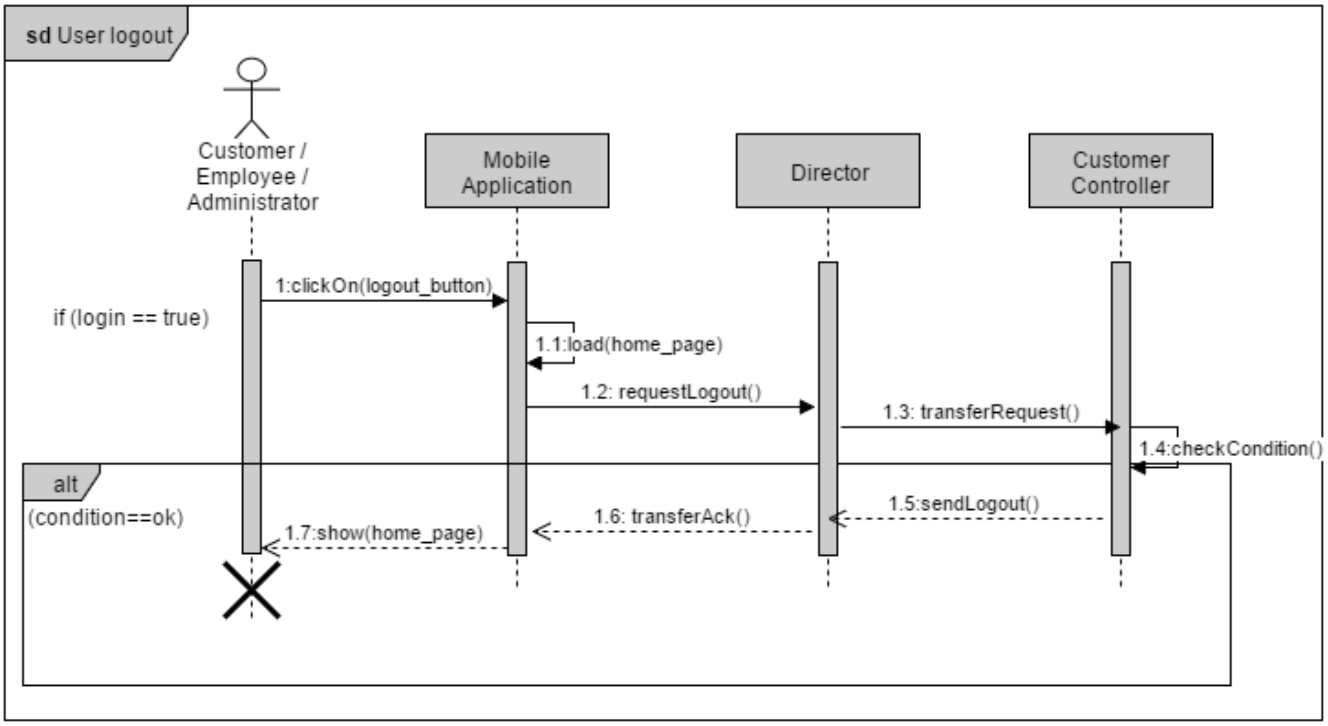

Fig. 2.8 Logout runtime view

If the customer wants to Logout (Fig. 2.8) has to click on the apposite button. If the request and the operation of disconnection from the server were successful, the customer will be redirected to the Home page.

2.5.4 Profile managing

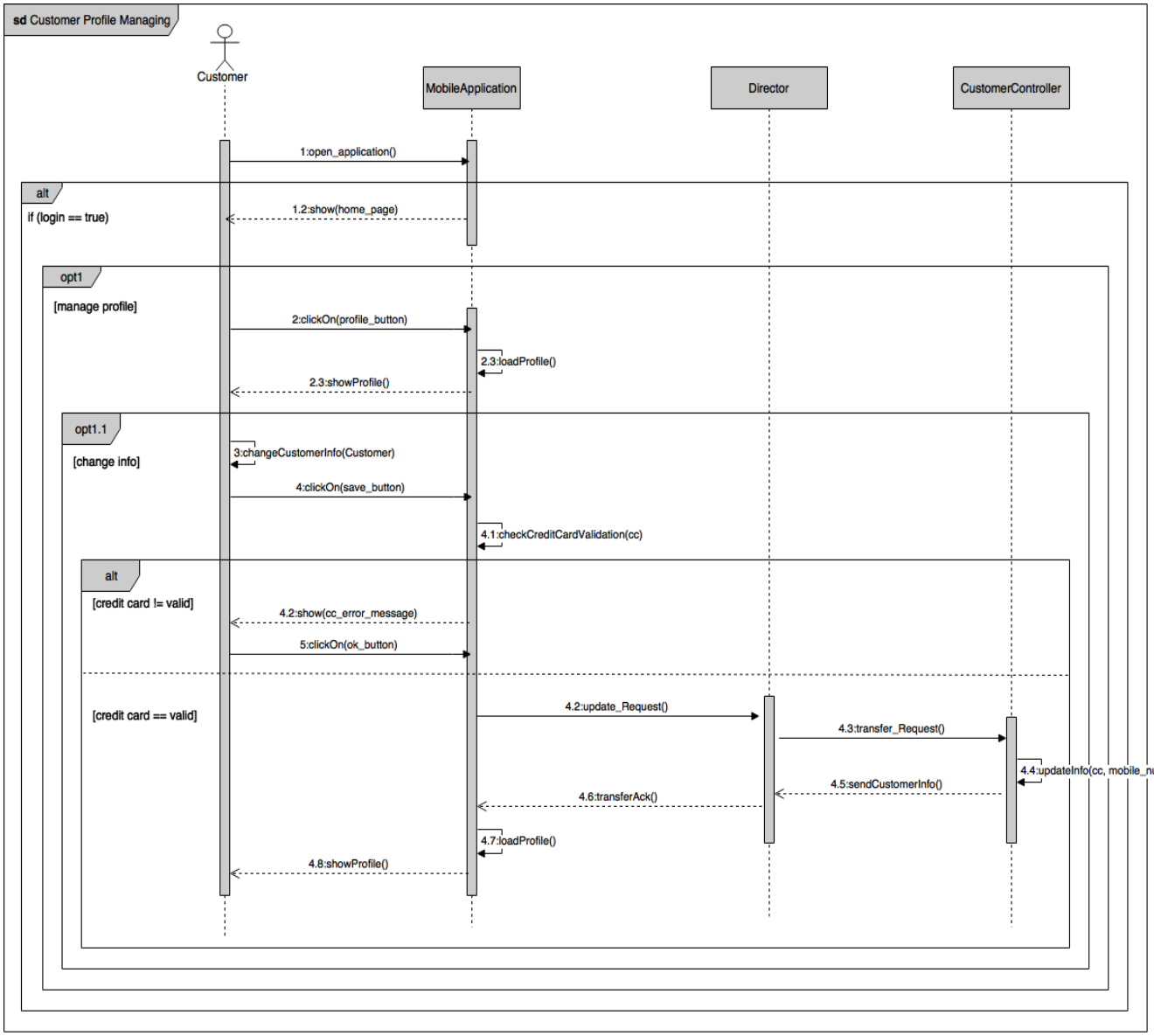

Fig. 2.9 Profile managing runtime view

The sequence diagram in Fig. 2.9 present the Profile Managing.

If the customer wants to change their information, he must enter the account panel. After the changes, he completes the request by clicking the button "save".

After the credit card validation, the change request is transferred to the CustomerController via the Director. At the end the Director send an acknowledgement to the Mobile Application in order to show the updated information the customer.

2.5.5 Car reservation

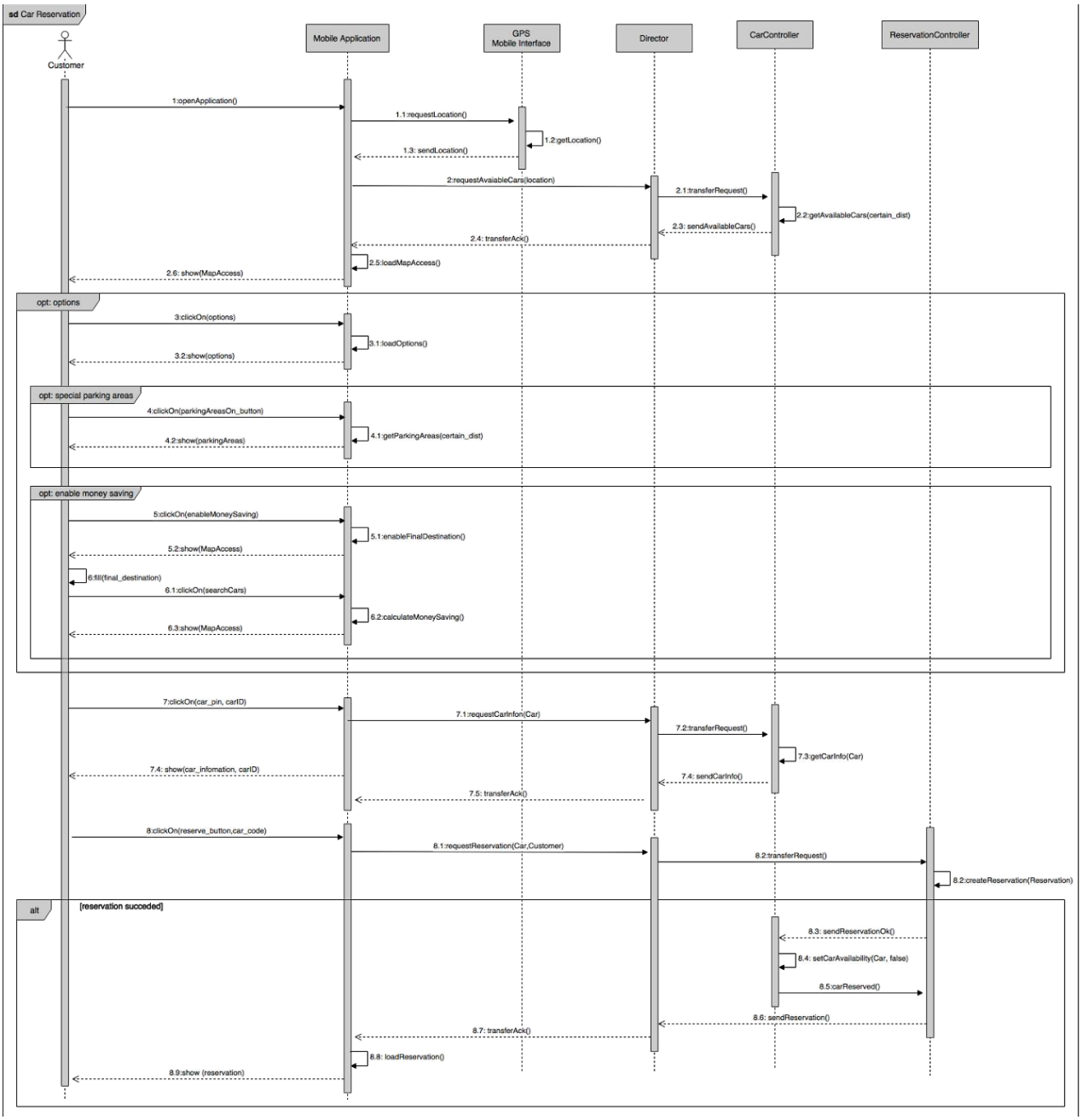

Fig. 2.10 Car reservation runtime view

Fig. 2.10 show Car Reservation runtime view. When customer opens the application in order to show the available cars on the map focusing nearby, a location request is send to GPS Mobile Interface to identify his/her location. This follows available cars request which is transferred to Director and handled by Car Controller. The available cars are send back to the Mobile Application which follows by it showing them on the Map Access page. If customer clicks on a car pin on the map, Mobile Application gets and sends this request as a Car Info request to the Director which routes it to the Car Controller. Car Controller gets the Car Info and direct it to Mobile Application through routing of Director.

Customer's click event on reserve button of the car is transferred to the Reservation Controller via Director as reservation request. Reservation Controller creates the requested reservation and if it is succeeded (satisfies the conditions), this controller makes change request of the car's availability to false to the Car Controller.

After the Car Controller performs this request, the Mobile Application informed by the acknowledgement which is send through Director.

2.5.6 Cancel reservation

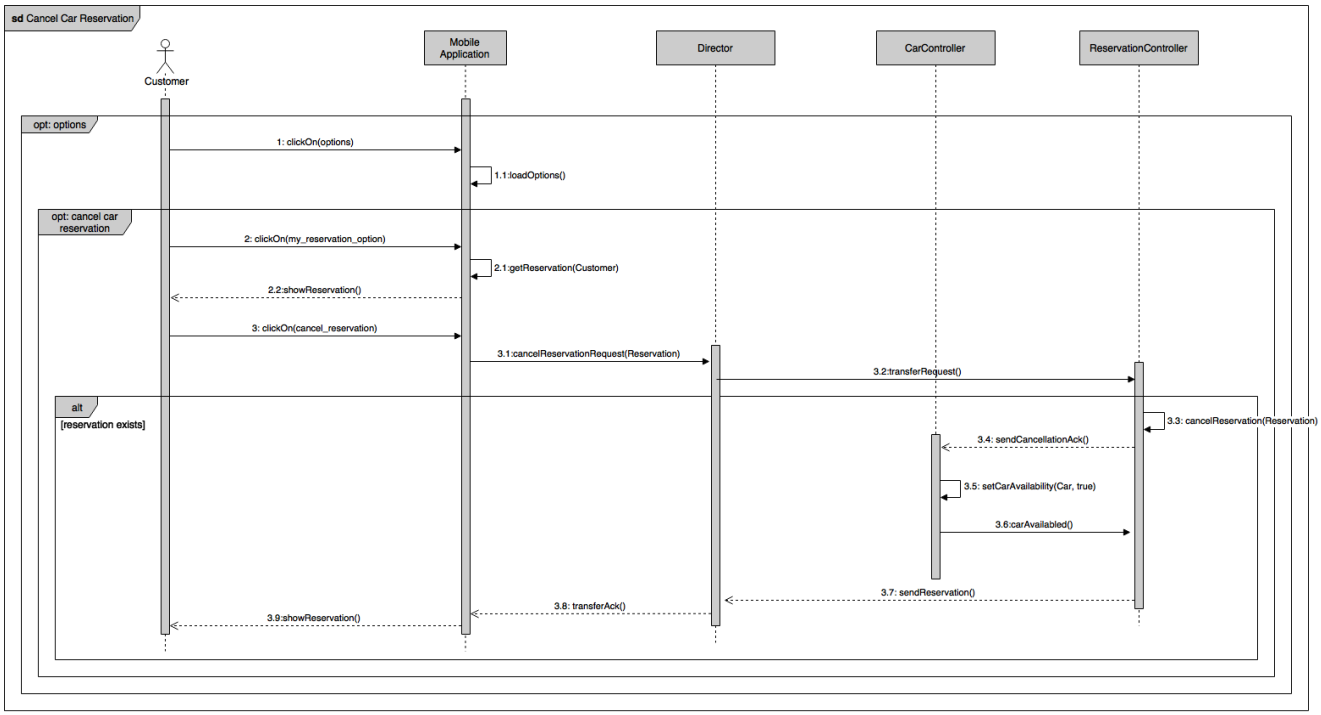

Fig. 2.11 Cancel Reservation runtime view

In the above sequence diagram of Cancel Car Reservation, it is seen that when customer requests a cancellation of his/her car reservation, this request sends to Reservation Controller via Director to perform this cancellation. Reservation Controller cancels the requested reservation and if it is succeeded (satisfies the conditions), this controller makes change request of the car's availability to true to the Car Controller. After the Car Controller performs this request, the Mobile Application informed by the acknowledgement which is send through Director.

2.6 Component interfaces

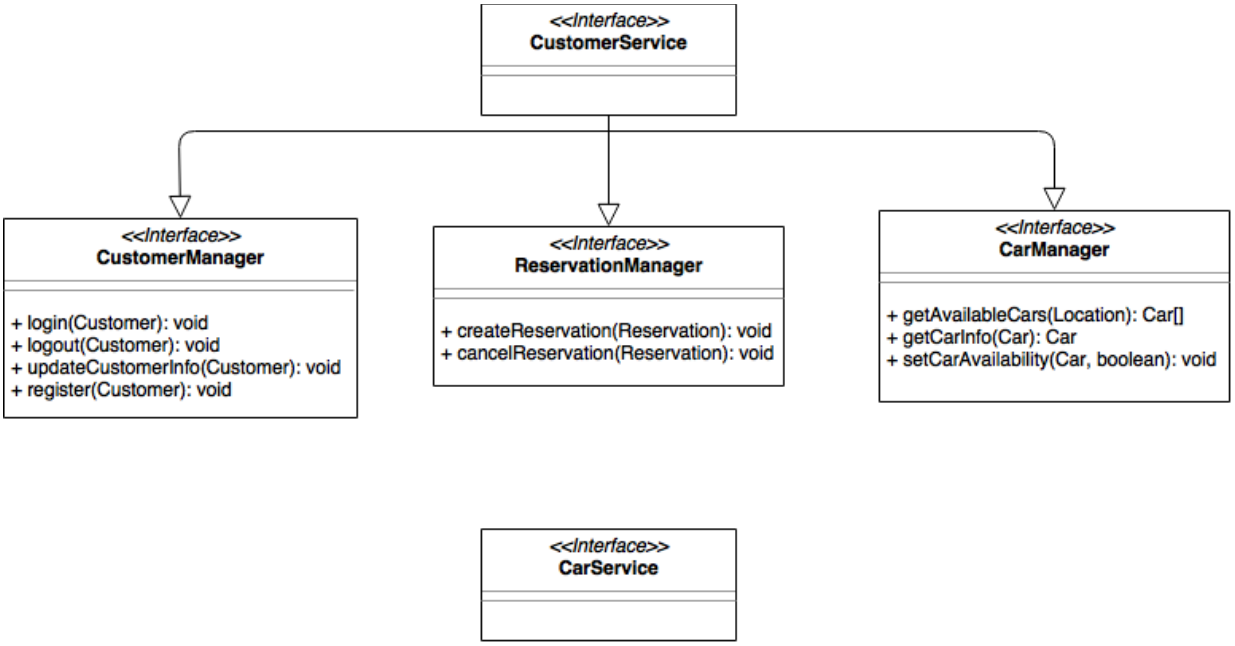

Fig. 2.12 Component interfaces

2.7 Selected architectural styles and patterns

2.7.1 Overall Architecture

The application is divided into:

- Data Layer
  (Data access and data store)
- Business Logic Layer
  (Application logic)
- Presentation Layer
  (Presentation and User interface)

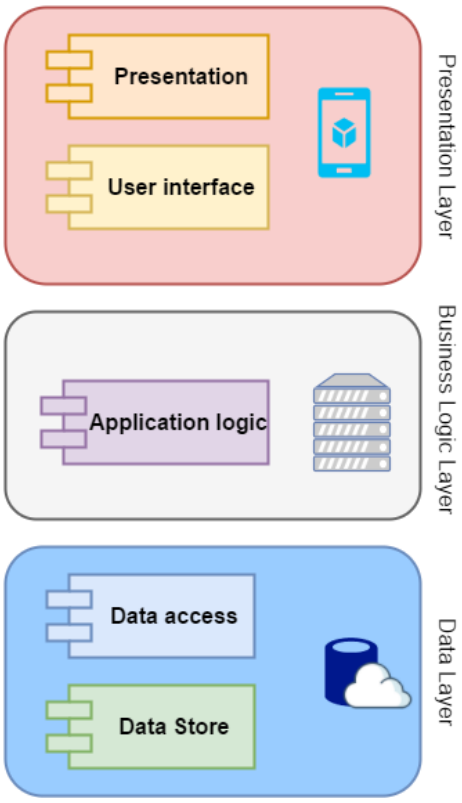

Fig. 2.13 Architectural Tiers

2.7.2 Protocols

JDBC: Java DataBase Connection used by Business logic layer to communicate with the Data layer. Here it report the major databases compatible with this protocol.

JDBC Driver Vendor	JDBC Driver Type	Supported Database Server
MySQL Connector/J Driver 5.1	Type 4	MySQL 5.1
Java DB 10.5.3.0	Type 4	Java DB 10.5.3.0
Oracle 11	Type 2 and Type 4	Oracle 11
PostgreSQL 8.4	Type 4	PostgreSQL 8.4
DB2 9.7	Type 2	DB2 9.7
Sun, DataDirect 4.0	Type 4	Sybase ASE 15
Sun, DataDirect 4.0	Type 4	DB2 9.7
Sun, DataDirect 4.0	Type 4	Microsoft SQL Server 2008
Sun, DataDirect 4.0	Type 4	MySQL 5.1

Fig. 2.14 DB supported bu JDBC

```
import java.sql.*;

public class ProvaJDBC {
  public static void main (String args[]){

    try {
      String driver = "sun.jdbc.odbc.JdbcOdbcDriver";
      Class.forName(driver);
      String url = "jdbc:odbc:myDataSource";
      Connection con = DriverManager.getConnection(url, "username", "password")
      Statement cmd = con.createStatement();
      String query = "SELECT * FROM nomeTabella";
      ResultSet res = cmd.executeQuery(query);
      while (res.next()) {
        System.out.println(res.getString("nomeColonna1"));
        System.out.println(res.getString("nomeColonna2"));
      }
      res.close(); // chiudere le risorse DB è obbligatorio
      cmd.close();
      con.close();
    }

    catch (SQLException e){
      e.printStackTrace();
    }

    catch (ClassNotFoundException e){
      e.printStackTrace();
    }
  }
}
```

Fig. 2.15 JDBC code example

2.7.3 Design patterns

MVC Model-View-Controller
It divides the application into three interconnected parts:

- A model stores data that is retrieved according to commands from the controller and displayed in the view.
- A view generates new output to the user based on changes in the model.
- A controller can send commands to the model to update the model's state. It can also send commands to its associated view to change the view's presentation of the model.

Client-Server
The server component provides a function or service to one or many clients, which initiate requests for such services.

2.8 Other design decisions

To integrate the mobile application with a map service it is determined to use an openmap service.

3 ALGORITHM DESIGN

3.1 Make a reservation

```
**********************************************************************************************************************
" CREATION OF CAR RESERVATION "
**********************************************************************************************************************

public boolean checkCurrentReservation(Customer cst){
// we check and return true if there exists a Current Reservation for the customer
}

public void createCarReservation(Location proposedDest, Car car, Customer cst){
    if(checkCurrentReservation(cst) == false)
    {
        CarReservation cr = new CurrentReservation;
        // set datas for the object attributes
        cr.customer = cst;
        cr.startingPoint = car.location;
        cr.reservationDate = LocalDateTime.now();
        if(msOpt) //if money saving is enabled
            cr.proposedDestination = proposedDest;
        car.availability = false;
        cst.history.Add(cr.reservationId, cr);
    }
}

public static Location getFinalDestination(){
    //Since we can use GoogleMaps API we are getting the input final destination like below.
    /* var input = document.getElementById('pac-input');
    var searchBox = new google.maps.places.SearchBox(input);
    map.controls[google.maps.ControlPosition.TOP_LEFT].push(input);

    // Bias the SearchBox results towards current map's viewport.
    map.addListener('bounds_changed', function() {
      searchBox.setBounds(map.getBounds());
    });

    var markers = [];
    // Listen for the event fired when the user selects a prediction and retrieve
    // more details for that place.
    searchBox.addListener('places_changed', function() {
      var places = searchBox.getPlaces();
      if (places.length == 0) {
        return;
    */
}


public static void calculateMoneySaving(){
    Location finalDest = getFinalDestination();
    //This station is determined to ensure a uniform distribution of cars in the city and depends both on the
    //destination of the user and on the availability of power plugs at the selected station
    proposedDest = //the station is determined
    showProposedDestinationMarker(proposedDest);
}

public static void showProposedDestinationMarker(Location proposedDest){
    //We show a marker for the calculated proposed destination
}
```

Fig. 3.1 Car Reservation Code

3.2 Calculate fees

```
**********************************************************************************************************************************
" CALCULATION FEE "
**********************************************************************************************************************************

float calculationFee (EndedReservation res){
    float cost= Costant.MINUTECOST;
    // Costant is a support Class to help the development

    float fee= (res.end.getMinute()-res.start.getMinute())* cost;
    // fee without discount or overcharge

    float discount= calculateDiscount(res, fee);

    return (fee-discount);
}


float calculateDiscount (EndedReservation res, float fee){
    float passengersDiscount=0;
    if (res.countPassenger()>= Constant.PASSENGERS)
    passengersDiscount= (fee*Constant.PASSDISC);

    float batteryDiscount=0;
    if (getBattery()>=Costant.MIDBATTERY)
        batteryDiscount=(fee*Constant.BATTDISC);

    float parkingDiscount=0;
    if (isPluged() || isInParkingArea(res.finalLocation))
        parkingDiscount=(fee*Constant.PLUGDISC);

    float overcharge=0;
    if(getPowerGridDistance()>=Constant.POWERGRIDDISTANCE || getBattery()<=Costant.LOWBATTERY)
        overcharge=(fee*Constant.OVERCHARGE);

    float moneySaving=0;
    if (res.finalDestination==getMoneySavingLocation())
        moneySaving=(fee*Constant.MONEYSAVDISC);

    return (passengersDiscount + batteryDiscount + parkingDiscount + moneySaving - overcharge);
}

boolean isInParkingArea(Location location){
    Map<String, Location> parkingAreas= getParkingAreas();
    for(String i: parkingAreasgetKey()){
        if (parkingAreas.getvalue(i)==location)
            return true;
        }
    return false;
}
```

Fig. 3.2 Calculate Fee Code

4 USER INTERFACE DESIGN

4.1 Mockups

The mockup document can be find in this GitHub link

4.2 UX diagrams

UX Diagram below is to show how the main actions that user (customer) performs.

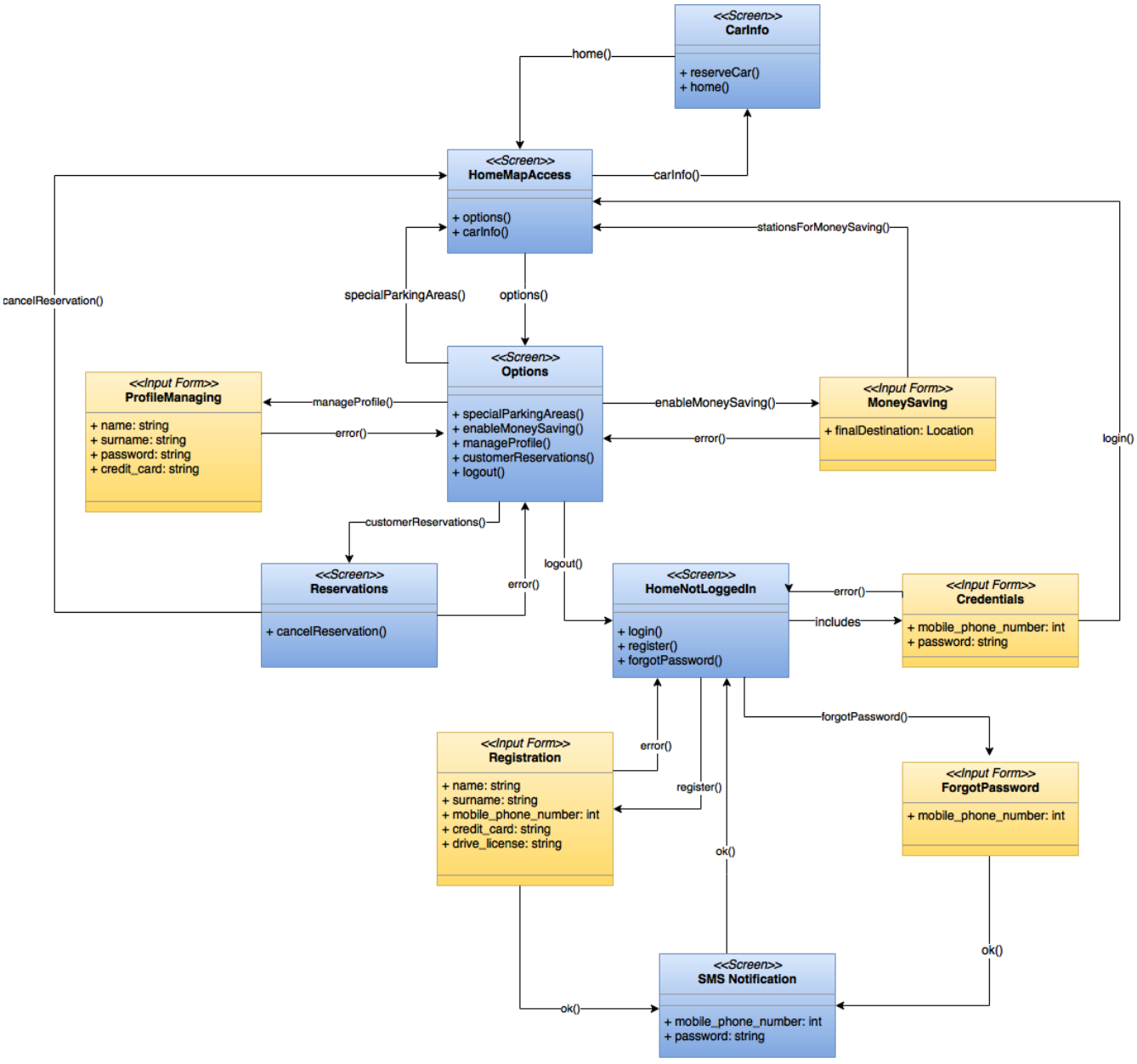

Fig. 4.1 UX Diagram

4.3 BCE diagram

BCE Diagram below is to show how user (customer) action is managed internally and how it's linked with the model. Since MVC is used in this project, it is crucial to point this out.

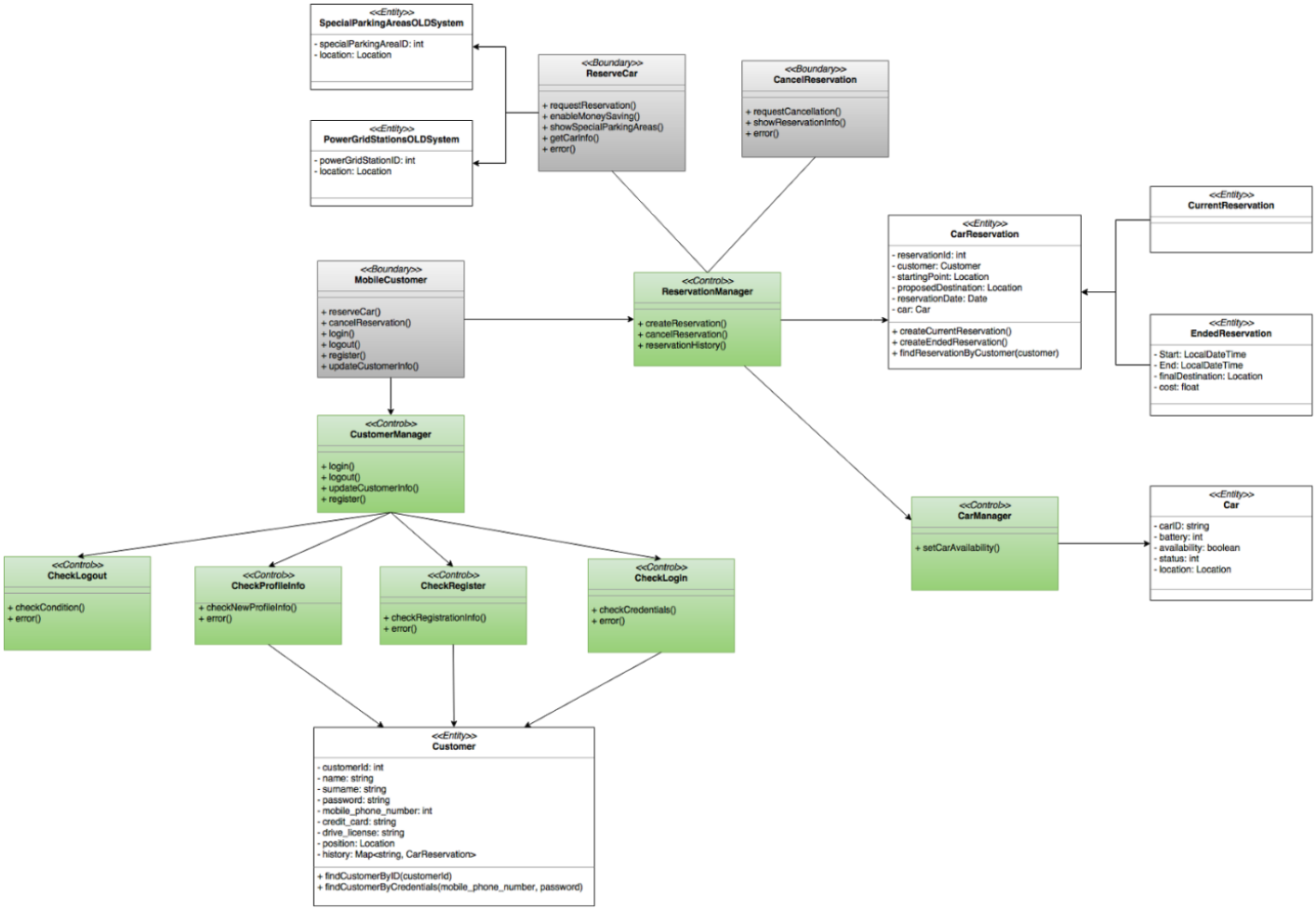

Fig. 4.2 BCE Diagram

5 REQUIREMENTS TRACEABILITY

The design of this project was made according to satisfy and fulfill the requirements and goals specified in the RASD. The requirement and goals listed below elaborates the design components that is designed to ensure their fulfillments.

- Registration to the system
  - Mobile Application
  - Director
  - Customer Controller
  - SmsSender
  - SmsGateway
- Login to the system
  - Mobile Application
  - GPS Mobile Interface
  - Director
  - Car Controller
  - Customer Controller
  - SmsSender
  - SmsGateway
- Log out from the system
  - Mobile Application
  - Director
  - Customer Controller
- Allowed to manage profile information
  - Mobile Application
  - Director
  - Customer Controller
- Allowed to reserve an available single car, in certain geographical region, for up to one hour before they pick it up.
  - Mobile Application
  - GPS Mobile Interface
  - Director
  - Reservation Controller
  - Car Controller
- Allowed to make a cancellation of the reservation
  - Mobile Application
  - Director
  - Reservation Controller
  - Car Controller
- Allowed to know the current charges through a screen on the car after system starts charging the customer for a given amount of money
  - Car Controller
- Allowed finding the locations of available cars within a certain distance from the current location of the customer, at the current time.
  - Mobile Application
  - GPS Mobile Interface
  - Director
  - Car Controller

- Allowed to list the properties (battery level, exact address, estimated distance can be made) of the available cars
    - Mobile Application
    - Director
    - Car Controller
- Allowed to see the safe area in the map that the car can be parked which is predefined by the management system
    - Mobile Application
    - GPS Mobile Interface
- Allowed to tell the system that customer is nearby so the system unlocks the car in order to enter
    - Mobile Application
    - GPS Mobile Interface
    - Director
    - Car Controller
- Allowed to see special parking areas to recharge the car by taking care of plugging the car into the power grid in order to get a discount.
    - Mobile Application
- Allowed to enable the many saving option, customer input final destination and the system provide the station information where to leave the car to get a discount.
    - Mobile Application

6 REFERENCES

http://www.mrwebmaster.it/java/gestire-connessioni-database-jdbc_10587.html

http://www.claudiodesio.com/java/jdbc.htm

https://it.wikipedia.org/wiki/Java_DataBase_Connectivity

6.1 Used tools

- Draw.io
- Pencil
- Microsoft Word
- Sublime text
- GitHub
- Google Drive
- Google Docs

7 HOURS OF WORK

Name & Surname	*29.11.16*	*01.12.16*	*04.12.16*	*10.12.16*	*Outside of Group*	*Total*
Burcu CESUR	2h	2h	2h	4h	27 h	37 hour
Marilena COLUCCIA	2h	2h	2h	4h	28 h	38 hour
Mustafa ÇEÇE	2h	2h	2h	4h	11 h	21 hour