Politecnico di Milano

Computer Science and Engineering's master degree course

PowerEnjoy
Design Document

Version 1.0

Release Date: 11/12/2016

Customer: Eng. Elisabetta DI NITTO

Authors:

Eng. Marco FERNI	Id. 877712
Eng. Angelo Claudio RE	Id. 877808
Eng. Gabriele TERMIGNONE	Id. 877645

Contents

Chapter 1

Introduction

A Purpose

This document extends the PowerEnjoy's RASD by providing a functional description of the system and more technical details. An overview of the system's design will be given through UML's Component, Deployment and Sequence diagrams, and we'll elaborate more on the User Interface part via the explanation of some already exposed mockups. The interactions between components, the deployment cycle and the runtime behavior of the system will be discussed, aiming to reach a level of description detailed enough for the software development to proceed with an understating of what are the software and hardware choices to be taken and how the system should be built.

B Scope

PowerEnjoy's main goal is to help people move around easier, without having to rely on their personal transport; a secondary goal is to reduction of cities' pollution and acoustic noise. To utilize Power Enjoy, a user need to successfully complete a registration procedure, in which he's asked to insert his IDs and driving licenses, together with some personal data. The system allows the now registered user to rent a car for a limited amount of time. The user can now start to look for a car by entering either his current position (detected by using their smartphone's GPS) or a specific location, chosen on the map, that he'll need to reach by himself. Frauds mechanism, like the 1 EUR fee, are applied to prevent abuse. The system policies encourage a smarter use of our service, by offering discounts to those who share a trip together. Users are also strongly suggested to leave a car in or near a PowerEnjoy's parking lot.

C Definitions, Acronyms, Abbreviations

- *Cost of the Trip*: raw price of the service calculated only on the base of the duration of the car's usage, before discounts or additional charges are applied.
- *Virtuousness coefficient*: the factor by which to multiply the cost of the trip to get the amount of the bill. Its initial value is 1.
- *Supervisor*: a company employee who work at the Car hub controller.
- *Recharge on site*: a company procedure: a worker is sent to recharge a low car that was parked detached from the power grid.
- *Car recovery*: a worker is sent to retrieve a car that has been forgotten outside a safe area and move that car into in one of these lot.
- *Guest*: a person who is not already registered to the system.
- *User*: a registered customer.
- *RASD*: Requirement Analysis and Specification Document
- *DD*: Design Document
- *DB*: Database
- *ER Diagram*: Entity Relation Diagram

D Reference Documents

- The PowerEnjoy's RASD (Requirement Analysis and Specification Document)
- The project's Assignments PDF
- Old years projects
- The DD standards
- Software Engineerign course slides

E Document Structure

The followind DD is composed in this way:

Chapter 2

Arhitectural Design

A Overview

A.1 General Structure

A Three-Tier architecture is used in this system:

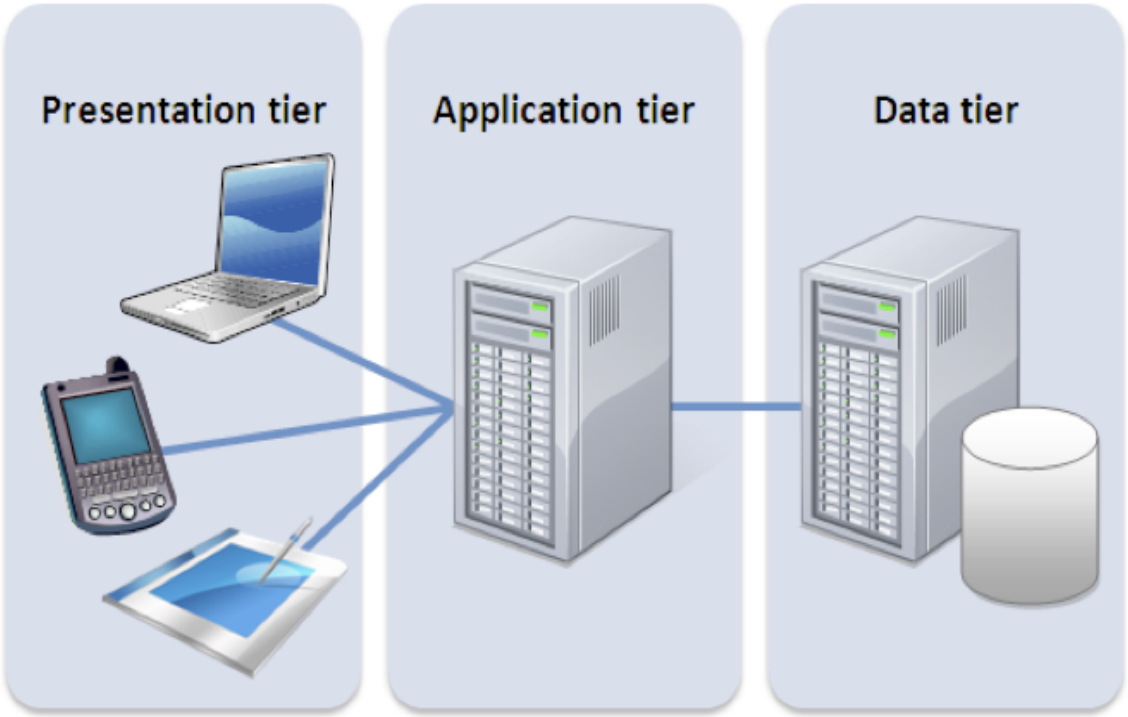


Figure 2.1: *Three Tiers Architecture*

More information about the chosen architectural style can be found in the section B.6.

A.2 High level components and their interaction

The following diagram gives a brief introduction of the system by showing the main components of the architecture and their basic relations:

- Central System
    - Application Server
    - DB
- Mobile App
- Car
    - Sensors
    - Display
- Supervisor

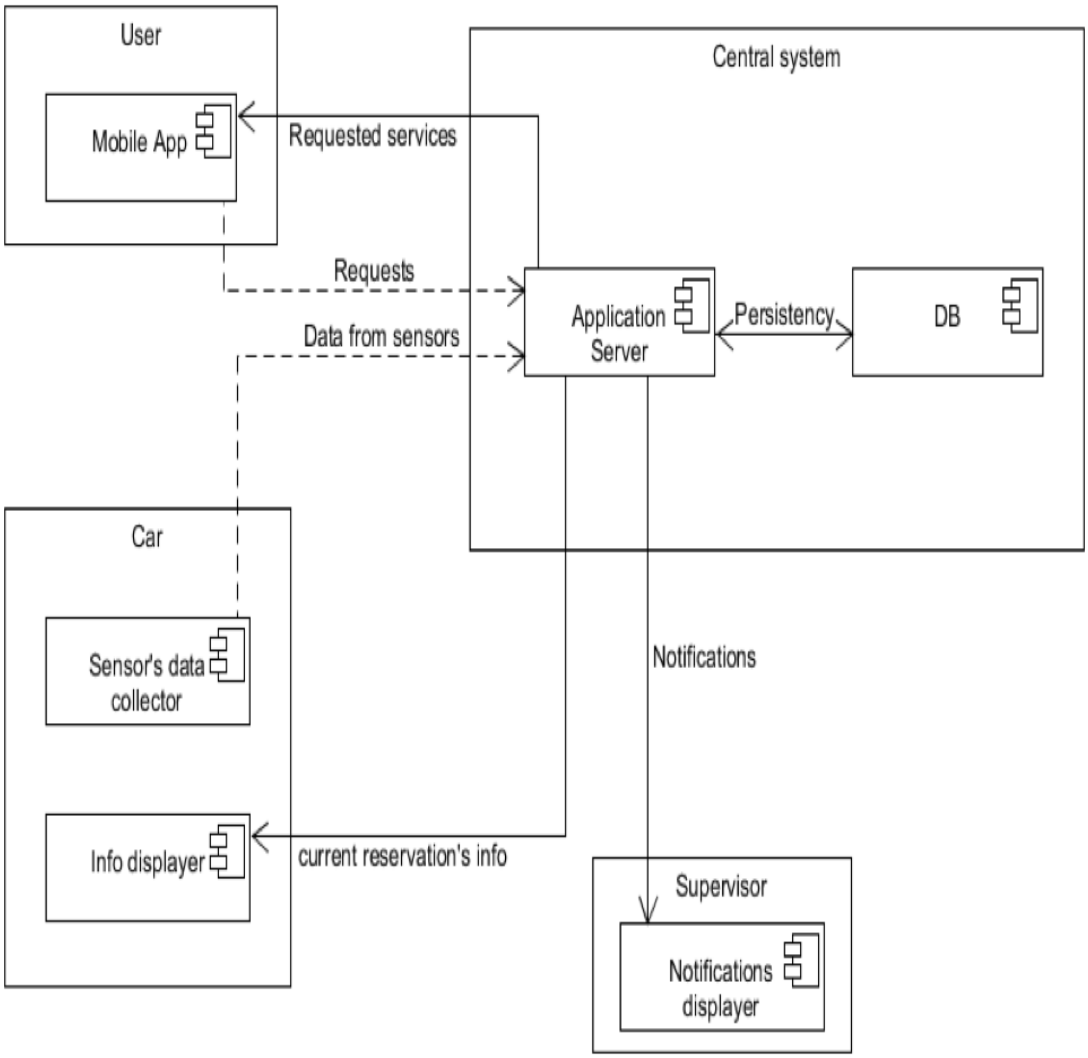


Figure 2.2: *High Level Architecture*

B Component view

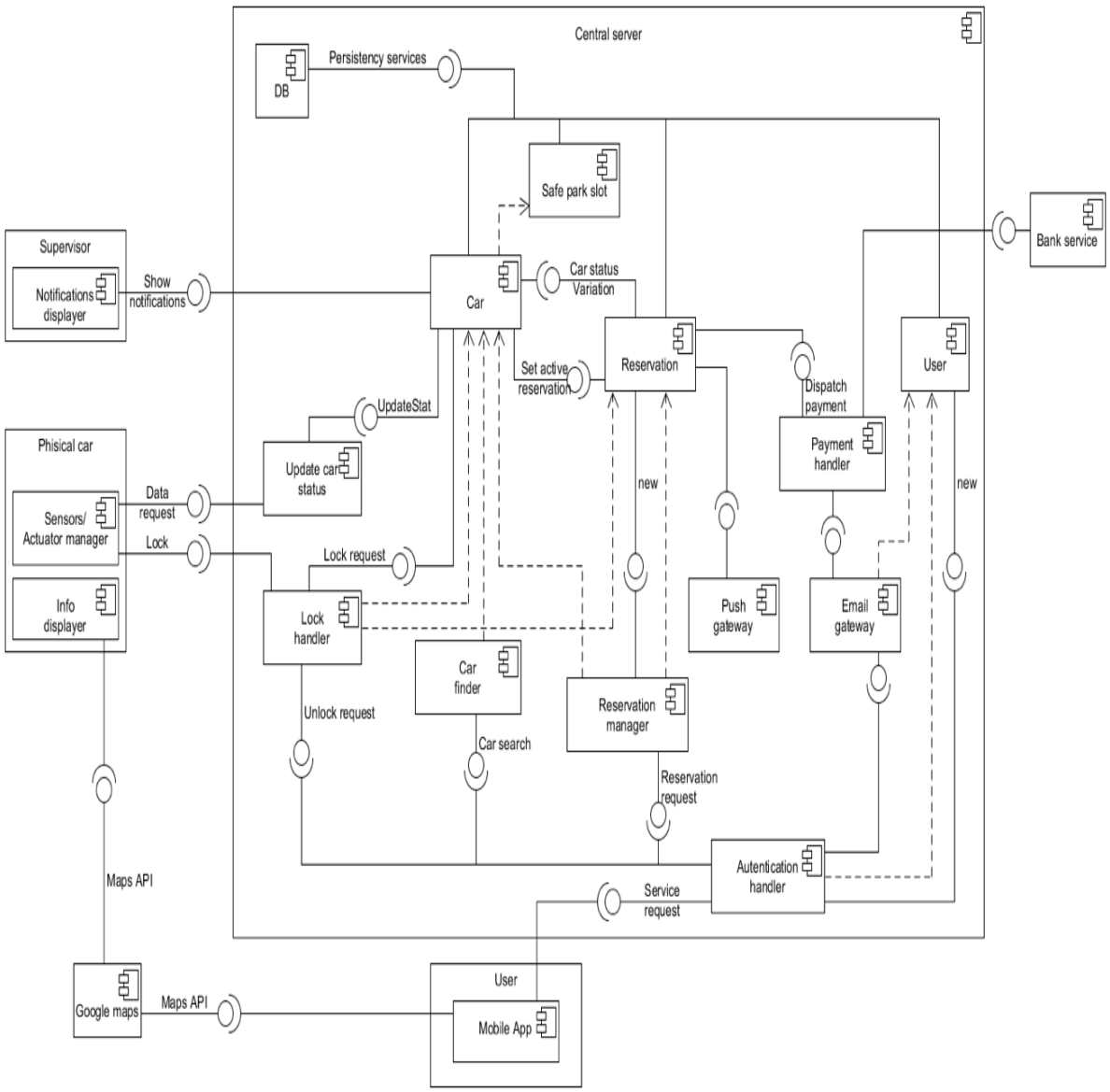


Figure 2.3: *Component Diagram*

NB: the dashed arrows represent the dependence of a component from the instances of the pointed part

B.1 DB Component and Interface

This component represents the Database used by the application server to store persistent data.

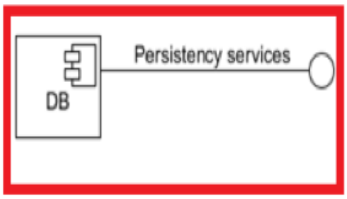


Figure 2.4: *component*

The Database's structure is explained in the following ER Diagram:

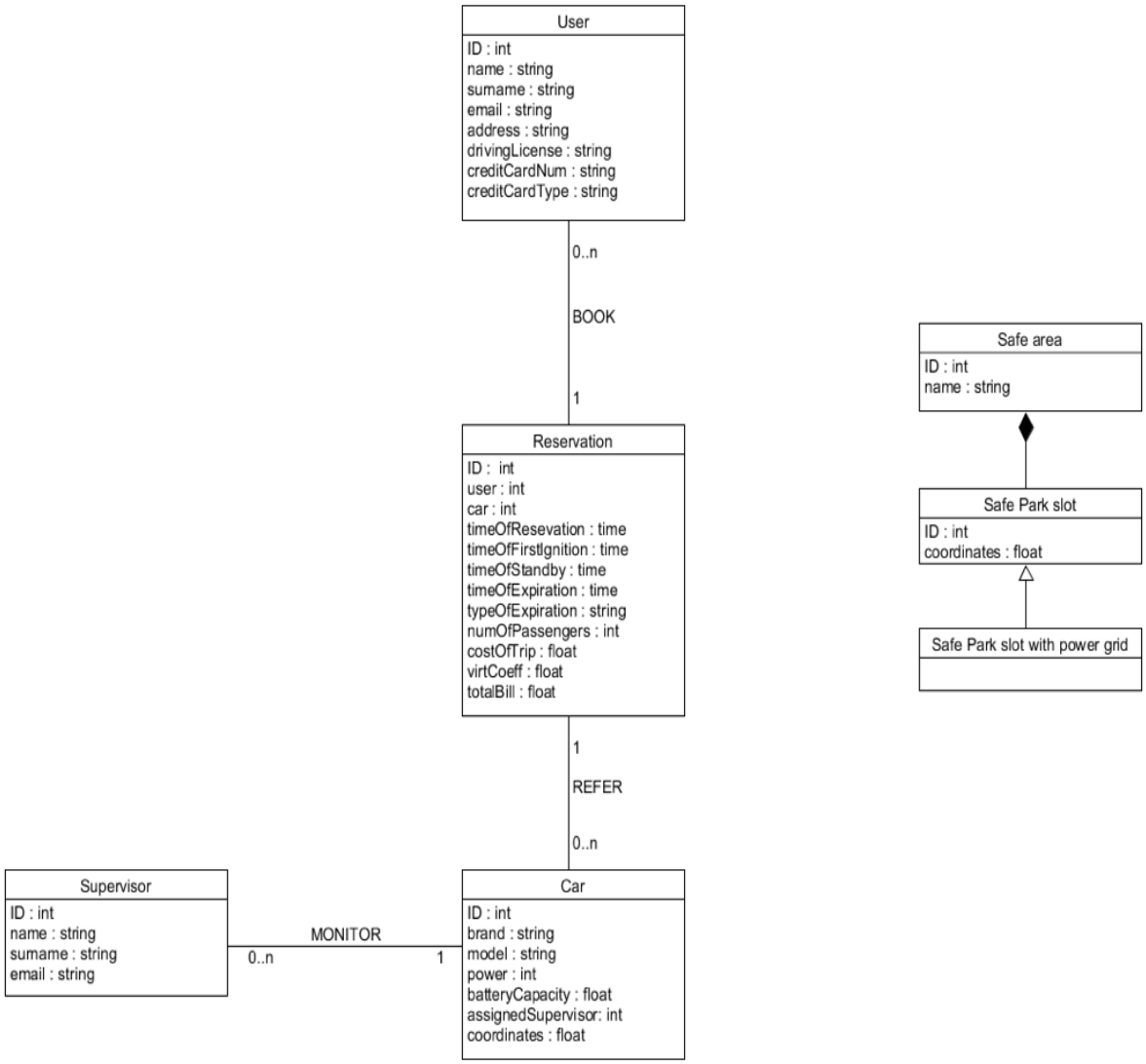


Figure 2.5: *ER diagram*

The interface "persistency services" represents all the methods that other components use to communicate with the database; these methods have to guarantee the atomicity, consistency and security of the transactions.

B.2 Safe park lot Component

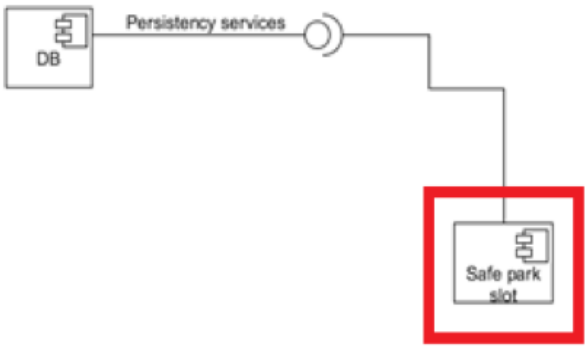


Figure 2.6: *component*

This component represents a simple entity bean; it's only meant to represent the entities "Safe park slot" of the database.

B.3 User Component and Interface

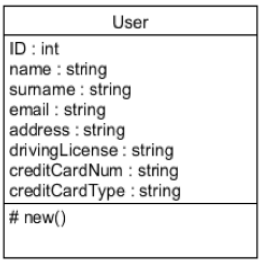


Figure 2.7: *class representation*

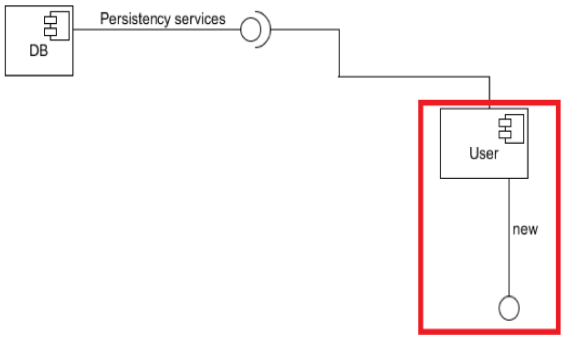


Figure 2.8: *component*

This component is an entity bean too. The "new" interface makes possible to create new "User" object, for example when a new customer registers to the application; it is up to the component to send the new information to the database.

B.4 Authentication handler component and interface

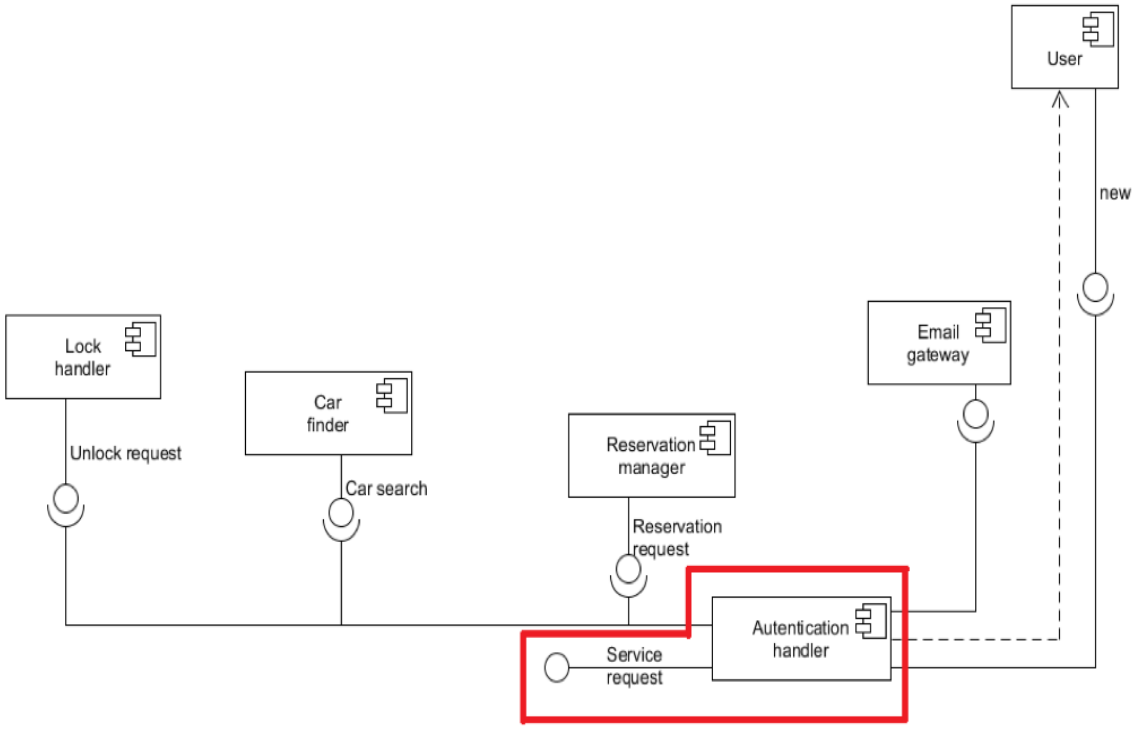


Figure 2.9: *component*

Authentication handler
sessions : session[] users : user[]
+serviceRequest() : obj -generateID() : int -generatePWD() : string -loginDataCheck() : bool -regDataCheck() : bool -isValidSession() : bool -createSession() : session

Figure 2.10: *class representation*

This component has the purpose of verifying the correctness of the data submitted by the user during the login phase, and to keep track of the user's session; another task of this component is to verify that each request coming from a user is made during a valid session: if this is true, the component manages to call the right method. The last task of this component is to manage the registration of

new users: after having verified the correctness of the data submitted, the "authentication handler" calls the "new" method of the component "user", and uses the "email gateway" component to send the confirmation email containing the password to the new user.

B.5 Reservation component and interface

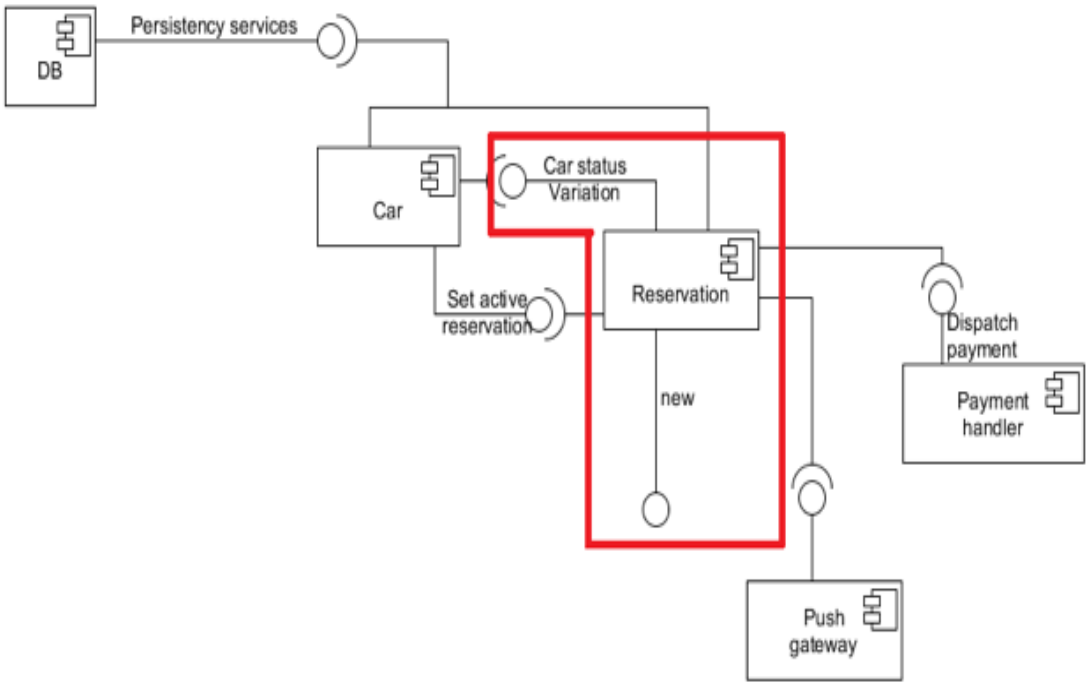


Figure 2.11: *component*

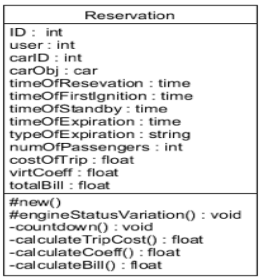


Figure 2.12: *class representation*

This component contains all the information about a certain reservation that are needed for the runtime functionality of the system. The "new" interface makes possible to create new "reservation" object. The method associated with this interface is responsible to initialize part of the object's attribute, to launch a 60 minutes' countdown, and to use the "set active reservation" interface of the

car object to which the reservation is associated. The "Car status variation" is used to notify the reservation object about the changing of specific attributes on 'state of the Associated car; this is useful because in this way the reservation can detect when the first engine's ignition occurs, so the system can register the number of passenger, or when a car is parked (in a safe area or not), in order to correctly launch the countdown, or ultimately, when a parked car is turned back on, in order to delete the running countdown. The private method "countdown" is used to determine when a reservation should expire: when it terminates, it sets the "timeOfExpiration" and "typeOfExpiration" attributes of the object, use the "set active reservation" interface of the car object in order to set the "actualReservation" attribute of the car to NULL, and launch sequentially the three method "calculateTripCost", "calculateCoeff" and "calculateBill"; after that, it calls the "payment handler" component. The last thing done by the method if it reaches the end of the countdown, is to send the information about the expired reservation to the database. The "push gateway" component is used to send specified information about the reservation to the user's app and to the car's monitor.

B.6 Car Component and interface

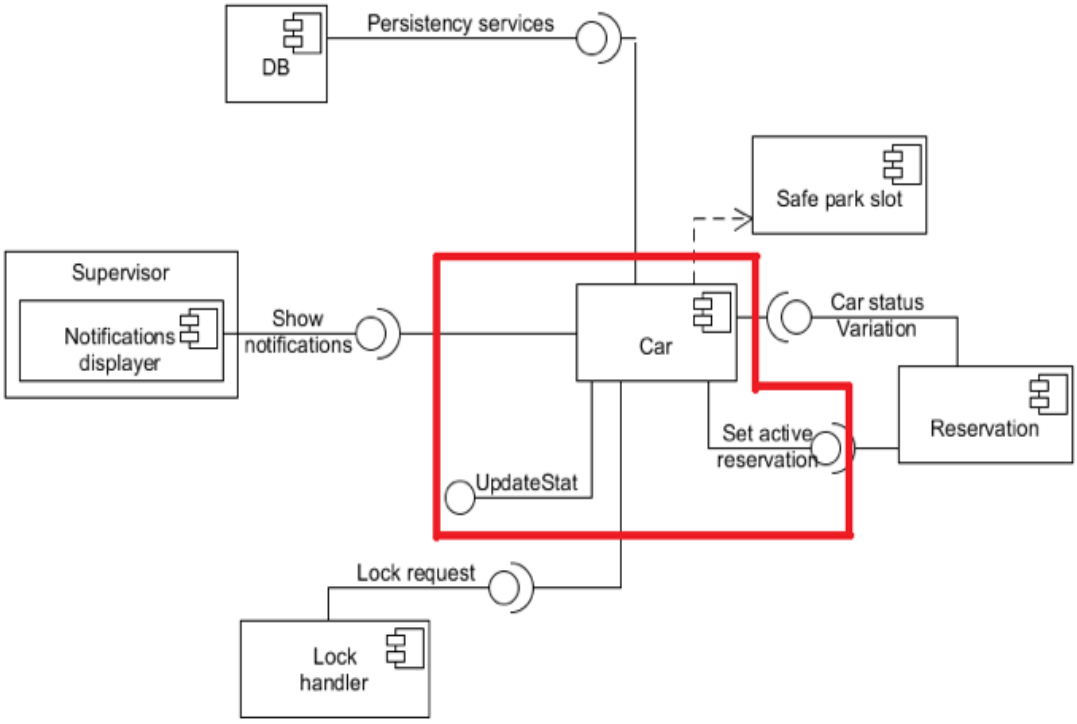


Figure 2.13: *component*

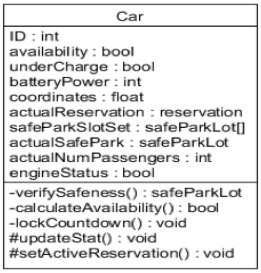


Figure 2.14: *class representation*

The "car" component represents the entity "car" of the database, but contains also other information that are needed for the runtime functionality.

The "updateStat" interface permits to an external component to keep the car object's attributes up to date; it is also responsible to call the "verifySafeness"

and "calculateAvailability" methods.

The "set active reservation" method is used by a new reservation to associate itself to the reserved car; it is also used by a reservation when it is about to expire, to set the "actualReservation" attribute of the associated car to null. It is also responsible to call the "calculateAvailability" methods.

When the methods associated with this interfaces are called, they perform various checks on updated data end eventually perform calls to another component:

Engine switched off and no passenger on board => car.lockCountdown + car status variation

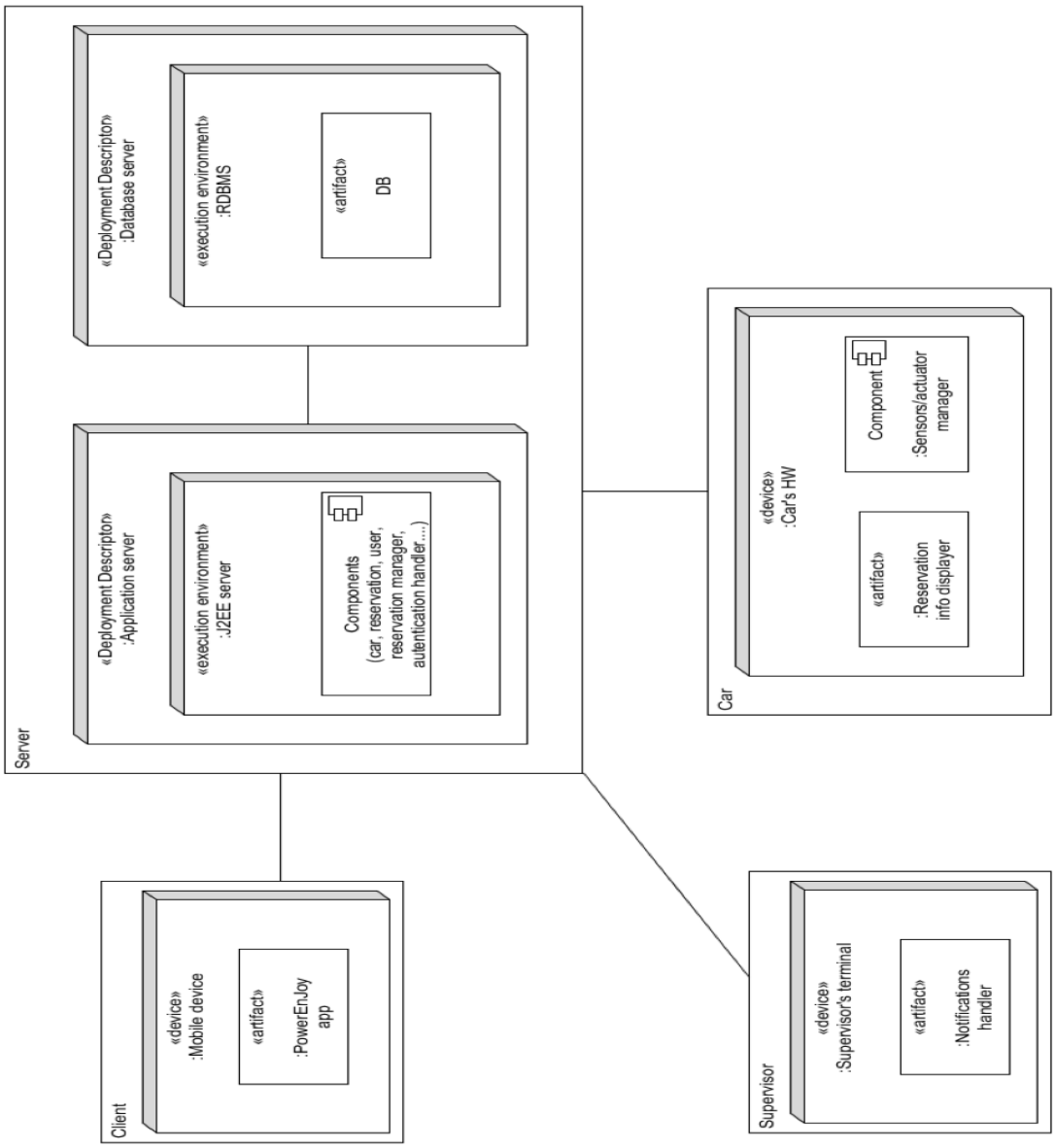
Engine switched on => car status variation

Reservation Expired && battery power < 20% && car not under charge => show notification

Reservation Expired && car is not in safe area => show notification

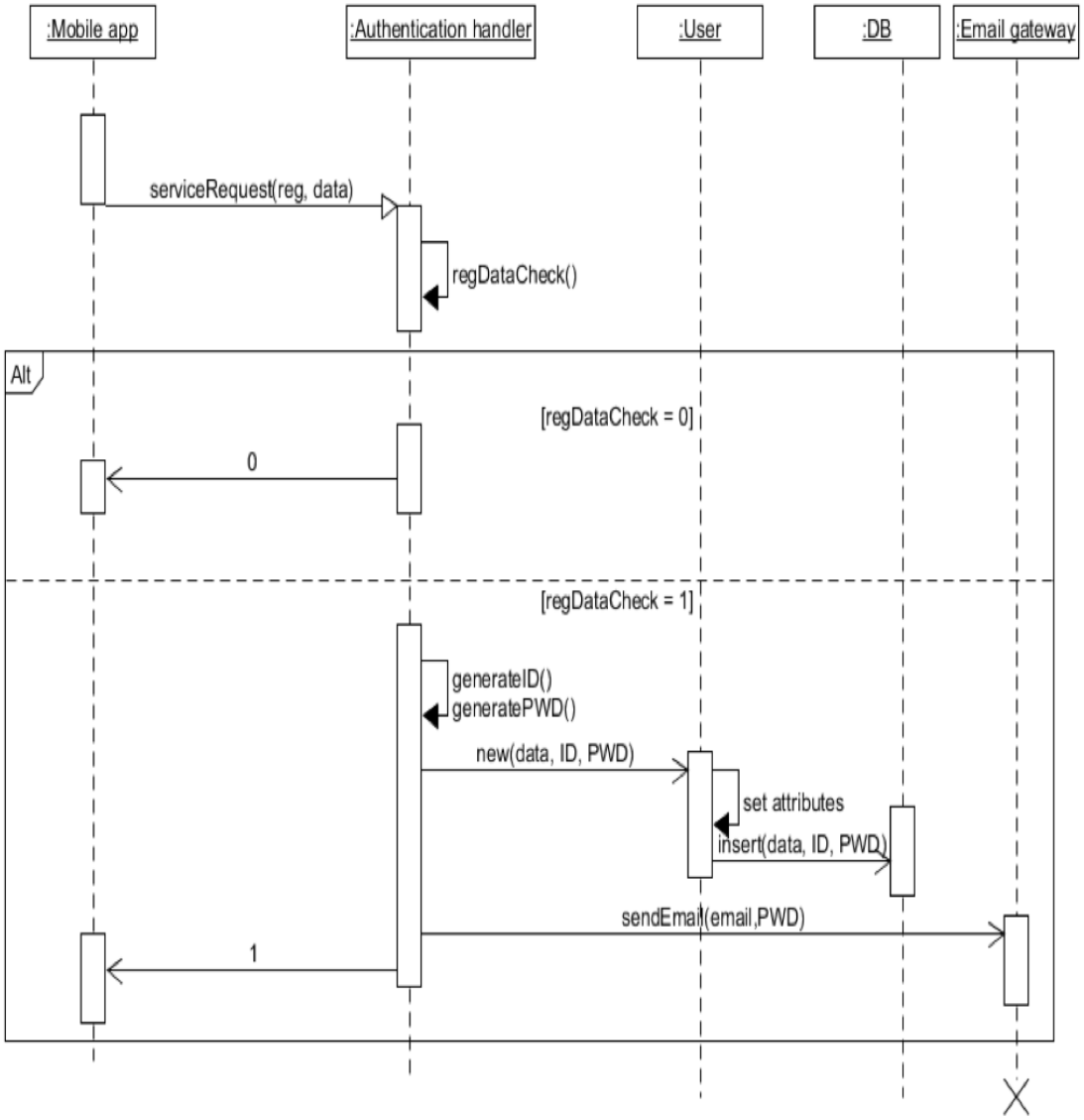
The "lockCountdown" method perform a 60 seconds countdown at the end of which it will be perform a lock request.

C Deployment view

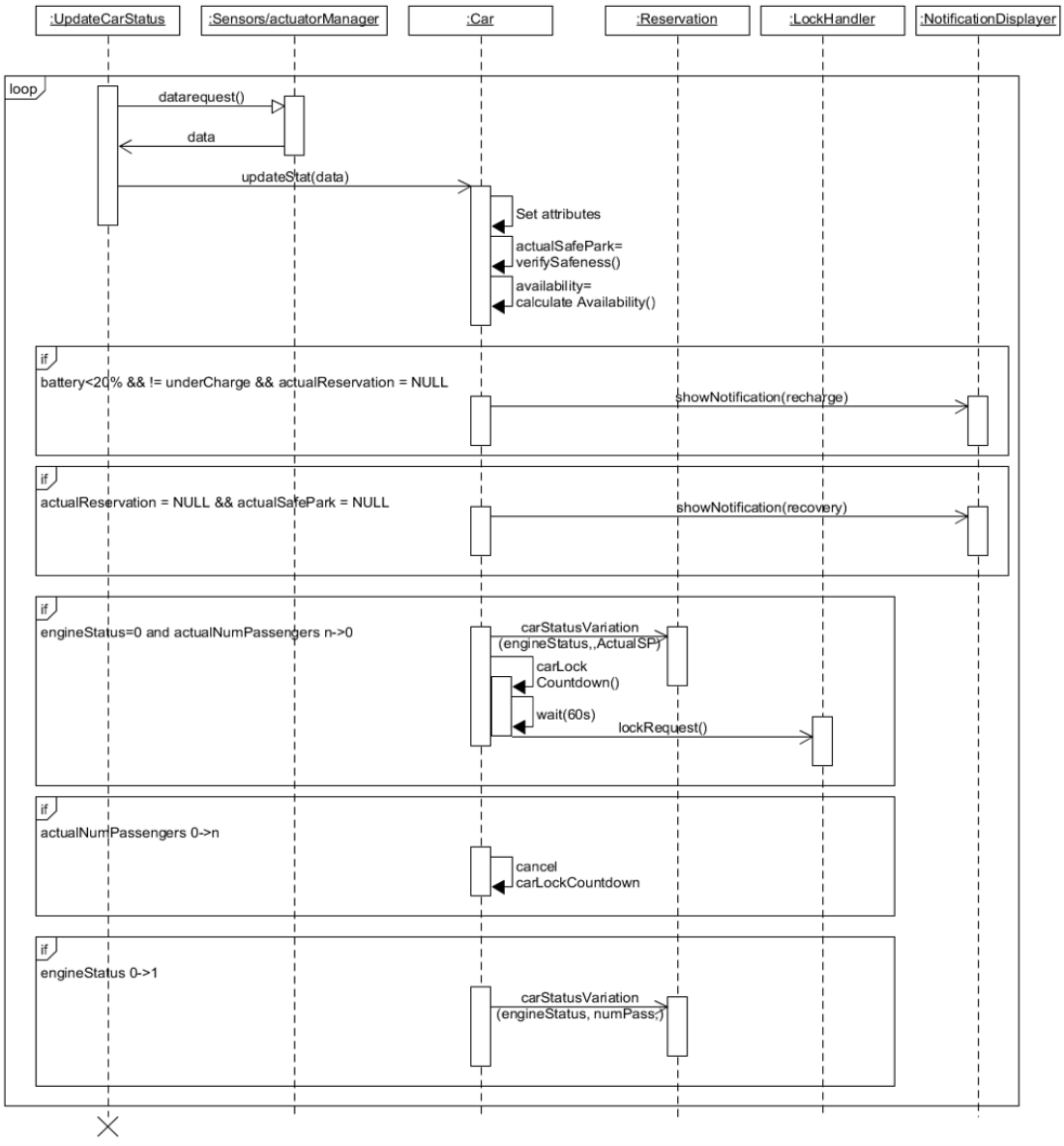


D Runtime view

D.1 Registration Sequence

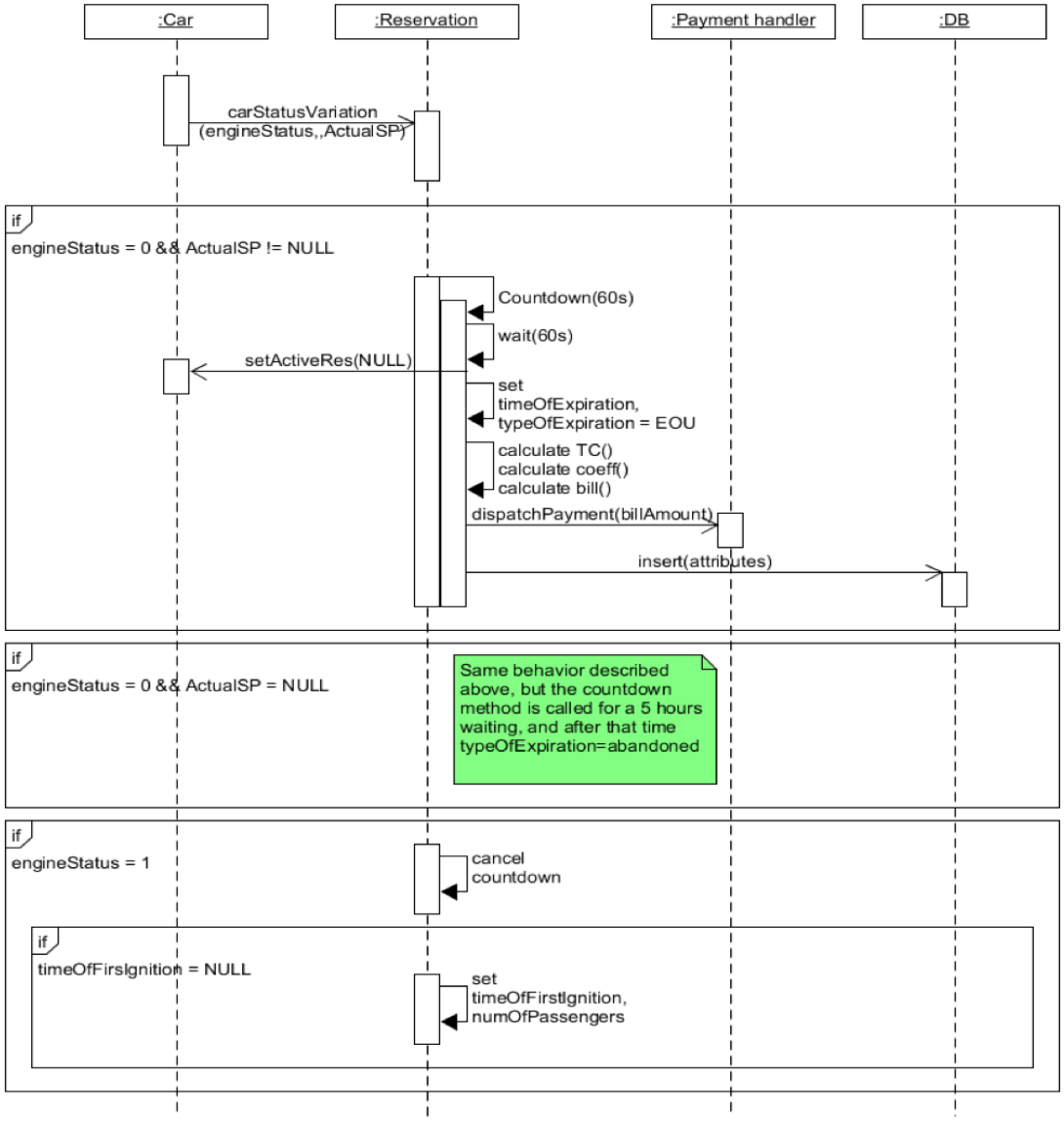


D.2 Update car status sequence



The "carStatusVariation" call will be explained in the next sequence diagram

D.3 Car status variation Sequence



This sequence diagram shows how variations of car's status influence the reservation

E Component interfaces

All the relevant interfaces have been already discussed in the section 2.2.

F Selected architectural styles and patterns

As previously introduced, a Three-Tier architecture is used in this project:

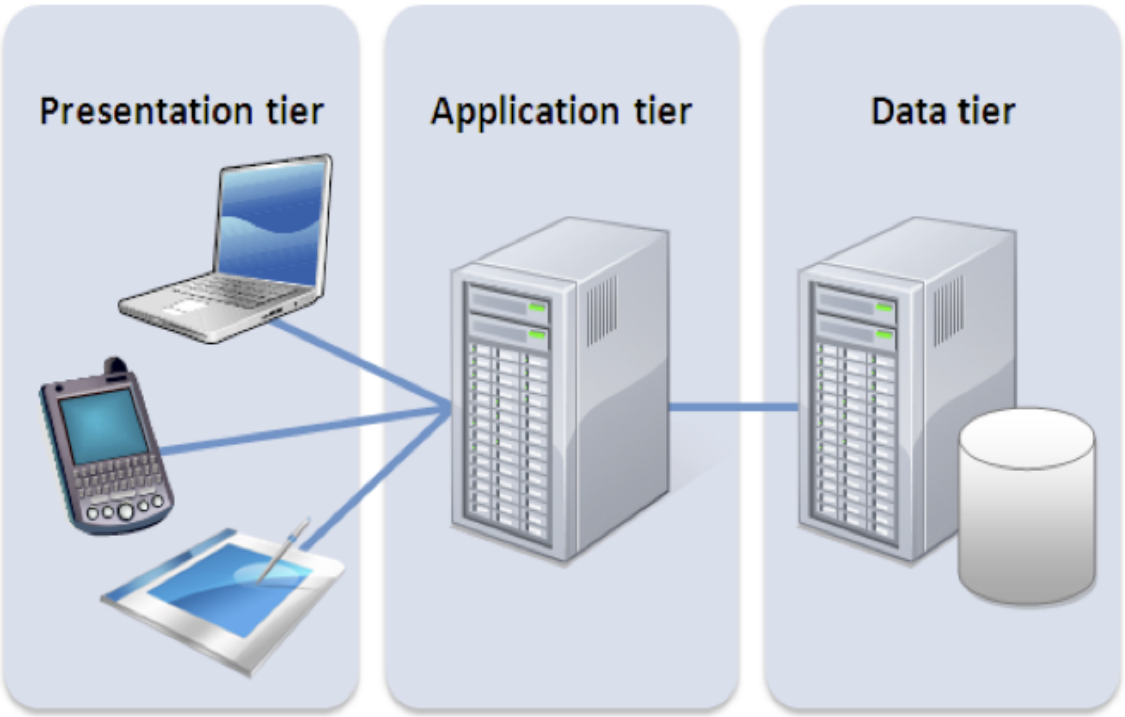


Figure 2.15: *Three Tiers Architecture*

1. Presentation Tier (Client Tier)
    a. Composed mainly by:
        i. The be the app used by the user to find and rent a car
        ii. The screen in the rented car and
        iii. Possibly, the tools used by the Supervisor.
    b. It's the layer that interacts with the server and displays useful information.
2. Application Server

   a. Where all the computations (reservations, locking/unlocking cars, payment) occur.

3. Data Tier

   a. MySQL DBMS

G Other design decisions

- The smartphone app will interact to the user via Push Notifications.
- All kinds of mobile devices are supported (Android, iOS and Windows Phone).
- Google Maps, with its API, is used in the app, allowing the user to:
  - locate a car (using the Google Maps' plotting capability) in the world before a reservation
  - o save money, when the saving option is chosen, thanks to the Route Suggestion Mode that draws an ideal path from his position to the parking lot minimizing his expenses.
- A premium GM's plan is expected to be used, due to the high volume of API requests needed for this app to correctly function.

Chapter 3

Algorithm Design

Chapter 4

User Interface Design

Chapter 5

Requirements Traceability

- [G1]: Ensure system's accessibility
- [G2]: Supervisors must be able to check cars' status
- [G3]: Supervisor should be able to dispatch "recharge on site" correctly
- [G4]: Supervisor should be able to dispatch "car recovery" correctly
- [G5]: Guarantee the correctness of each car's "availability state"
- [G6]: Allow user to find available cars within a certain distance from a specified place
- [G7]: Allow user to reserve a single car
- [G8]: Discourage fake and too long reservation
- [G9]: Allow the user who reserved the car to see information about his reservation
- [G10]: Allow only the user who reserved the car (and his passenger) to access it
- [G11]: Guarantee the correctness of the "cost of the trip"
- [G12]: Guarantee the correctness of the "virtuousness coefficient"

- [G13]: Guarantee the correctness and the payment of the final bill
- [G14]: Support the users saving money

GOAL	**Component**	**Entity**
G1	Mobile App, Authentication handler	User, Central server
G2	Notifications displayer, Car	Supervisor, Central Server
G3	Notification displayer	Supervisor
G4	Notification displayer	Supervisor
G5	Update car status, Car, Reservation, Safe park slot, Sensor/actuator manager	Central server, Physical car
G6	Mobile app, Authentication handler, Car finder, Google maps API	Central server, User, External
G7	Mobile app, Authentication handler, Reservation manage Reservation	Central server, User
G8	Reservation manager	Central server
G9	Mobile app, Authentication handler, Reservation manager, Reservation	Central server, User
G10	Mobile app, Authentication handler, Lock handler, Sensor/actuator manager	Central server, User, Car
G11	Reservation, Car	Central server
G12	Reservation, Car	Central server
G13	Reservation, Car Payment handler	Central server
G14	Info displayer, Google maps API	Car, External

Chapter 6

Effort Spent

Date	Marco (h)	Angelo (h)	Gabriele (h)
16/11/2016		0.5	
21/11/2016			1
22/11/2016			1
29/11/2016	6	4	4
30/11/2016	5		3
01/12/2016	0.5		6
02/12/2016	5		5
03/12/2016	1	1.5	
04/12/2016	5		
06/12/2016	3	3	3
08/12/2016	3	5	
TOTAL	28.5	14	23

Chapter 7

References

A Tools Used

- Sketch: for the mokups' graphical rapresentation
- MS Word: as a shared text editor
- Github: to publicize our work
- MikTeX
- TexMaker
- Photoshop
- Uml Drawer

B ChangeLog

Version 1.0 - First Release Version