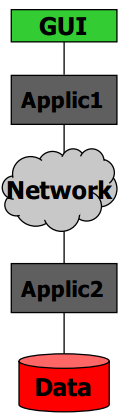
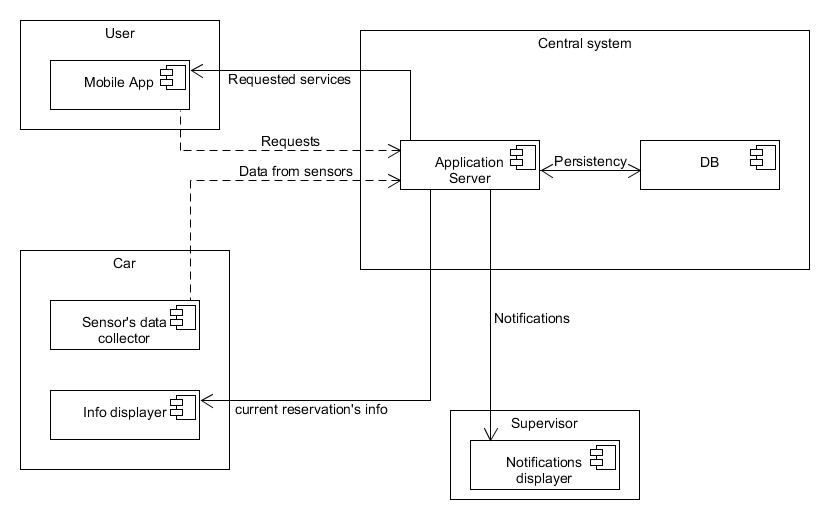
**2.1 Overview**

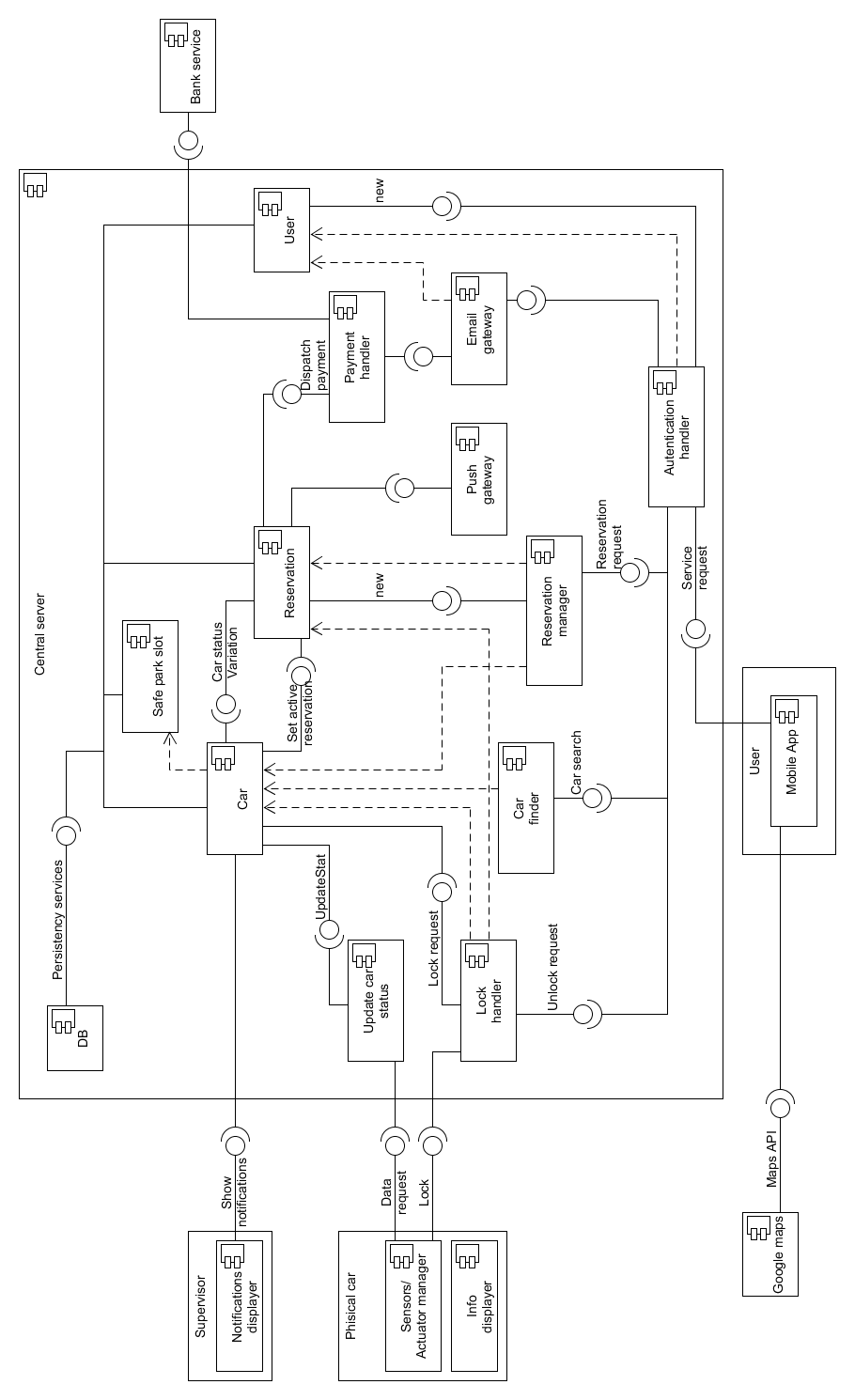
**2.1.1 General structure**

2 level C/S tier (distributed logic)

****

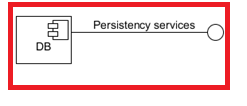
**2.1.2 High level components and their interactions**



**2.2 Components:**

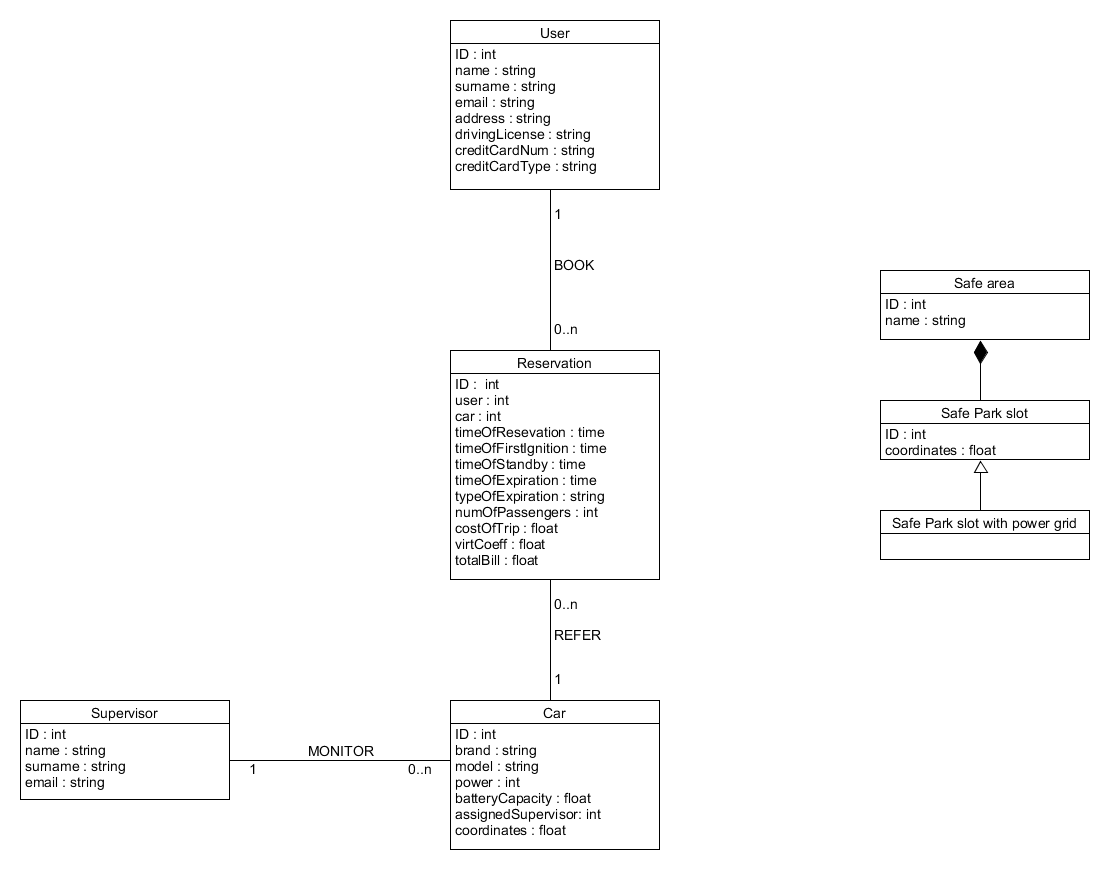
*Component diagram*

**2.2.1 DB Component and interface**

****

*Component*

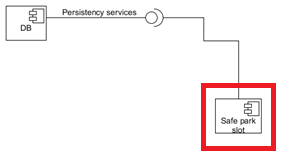
The component represents the Database used by the application server to store persistent data.



*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.

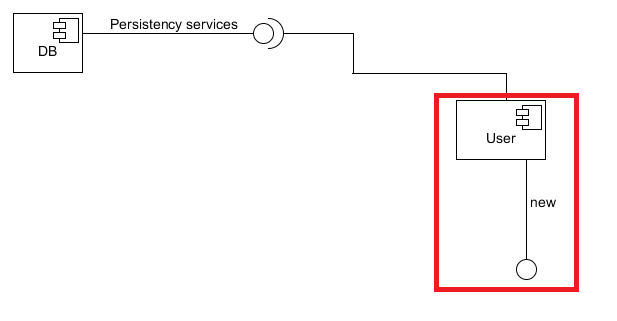
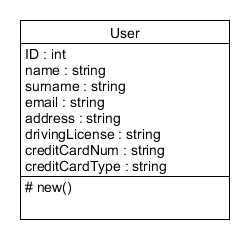
**2.2.2 Safe park lot Component**



*Component*

This component represents a simple entity bean, it’s only meant to represent the entities “Safe park slot” of the database

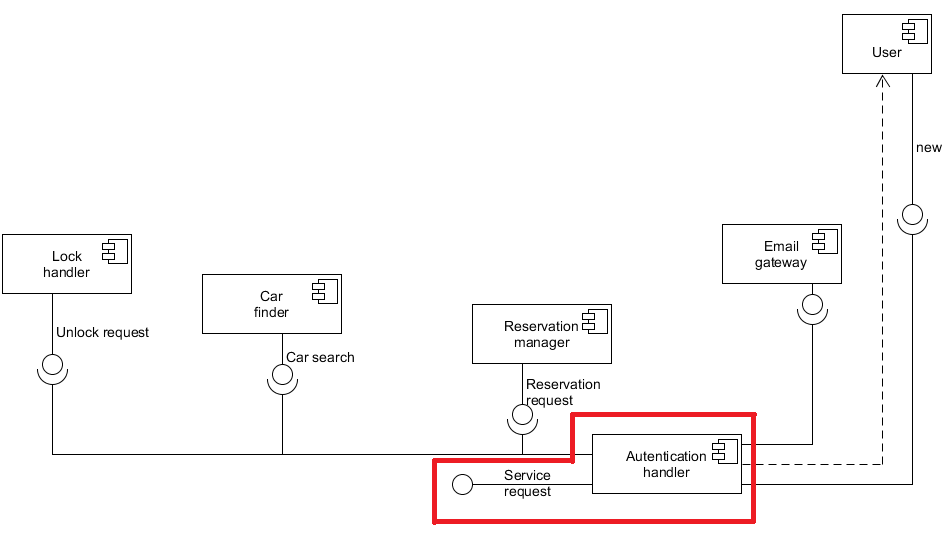
**2.2.3 User Component and interface**



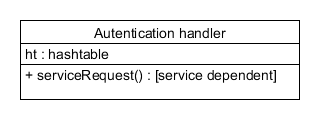
*Component class representation*

This component is an entity bean too; in addition, 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.

**2.2.4 Autentication handler component and interface**



*Component*

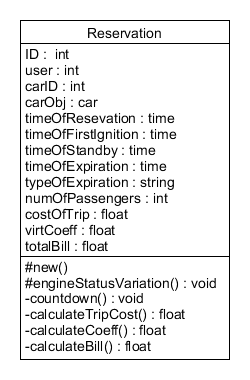


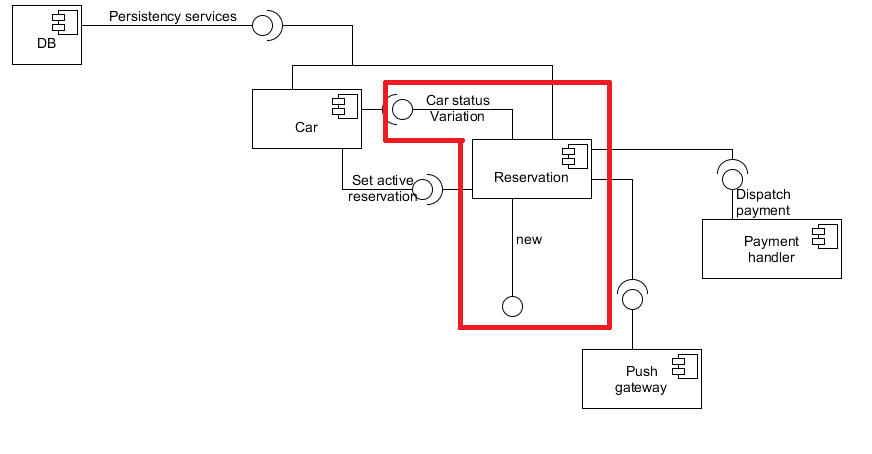
*class representation*

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.

**2.2.5 Reservation component and interface**



****

*Component class representation*

This component contains all the information about a certain reservation that are needed for the runtime functionality of the system.

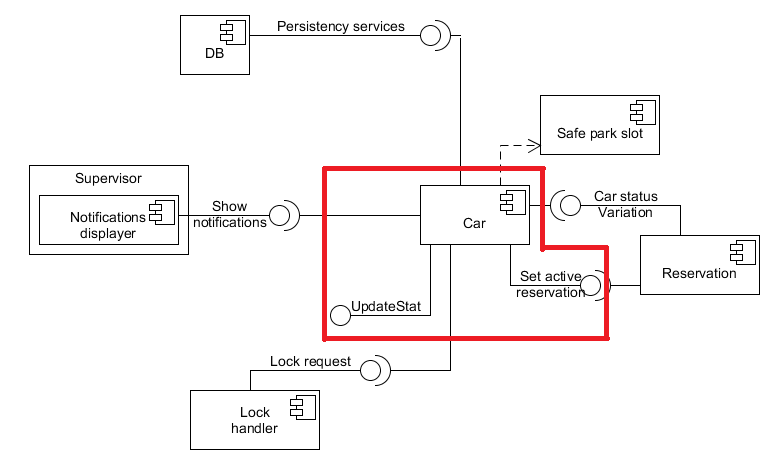
The “new” interface makes possible to create new “reservation” object. The method associated with this interface is responsible to initialize part of the object’s attribute, to launch a 60 minutes’ countdown, and to use the "set active reservation" interface of the car object to which the reservation is associated.

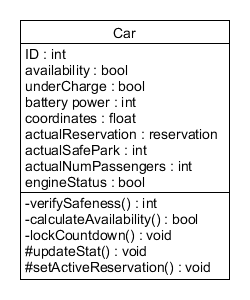
The "Car status variation" is used to notify the reservation object about the changing of specific attributes on 'state of the Associated car; this is useful because in this way the reservation can detect when the first engine’s ignition occurs, so the system can register the number of passenger, or 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 “timeOfEpiration” 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.

**2.2.6 Car component and interface**



*Component 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.

Ore di lavoro:

21/11/16: 1h

22/11/16: 1h

29/11/16: 4h

30/11/16: 3h

01/12/16: 6h