**2. Integration Strategy**

**2.1. Entry Criteria**

This section describes the prerequisites that need to be met before integration testing can be started.

First of all, code inspection has to be performed on all the code in order to find possible issues and to ensure maintainability and respect of conventions.   
Taking as reference the component diagram (2.2 from DD), the following component must be unit-tested before our integration tests:

• Notifications displayer (Supervisor)  
• Sensor/actuator manager (Car)  
• Info displayer (Car)  
• Mobile App (User)  
• DB  
• Email gateway  
• Push gateway

The DB component represents the database: because the database run on the same system of the application server, in the component diagram it is represented as a simple component; following this convention, we will continue to represent it as a component and to consider it as part of the “central server” subsystem.

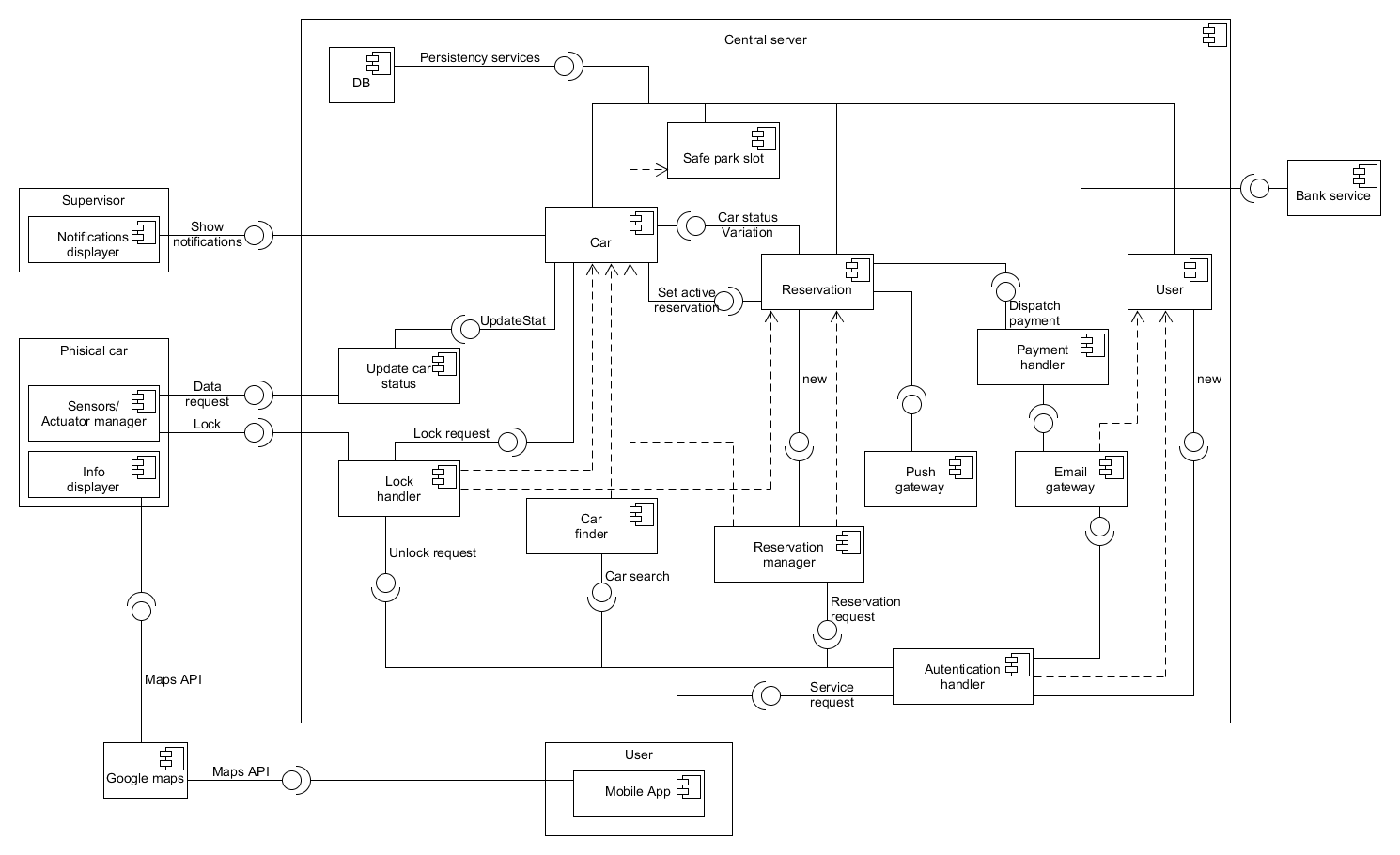
In addition, there are present in the diagram 2 components which represent external services (Google maps and bank service), which can be considered already reliable and do not need to be unit tested.

We should test all non-trivial methods, getter and setter methods can be skipped.

The components “User” and “Safe park slot”, which contain only getter and setter methods, are considered as already tested.

**2.2. Elements to be integrated**

The elements to be integrated are all the elements represented in the already mentioned component diagram, considering that our system must cooperate with 2 external services (google maps and the bank service).

****

*Component diagram*

**2.3. Integration testing strategy**

The integration strategy that we decided to follow is the bottom-up approach. The main reason for this choice is that we assume that we already have the unit-test for some of the simplest components, so we can proceed from the bottom.

Moreover, except for the central server, the other high-level components are composed by simple parts that we consider as atomic; the high level components communicate through well defined interfaces (REST API), so the integration of each of them will not be hard in a later time.

In addition, this approach has other intrinsic advantages: we can limit the usage of stubs, the errors are well located and, if the realization of the components follows a bottom-up approach too, the testing of lower level modules can take place earlier.

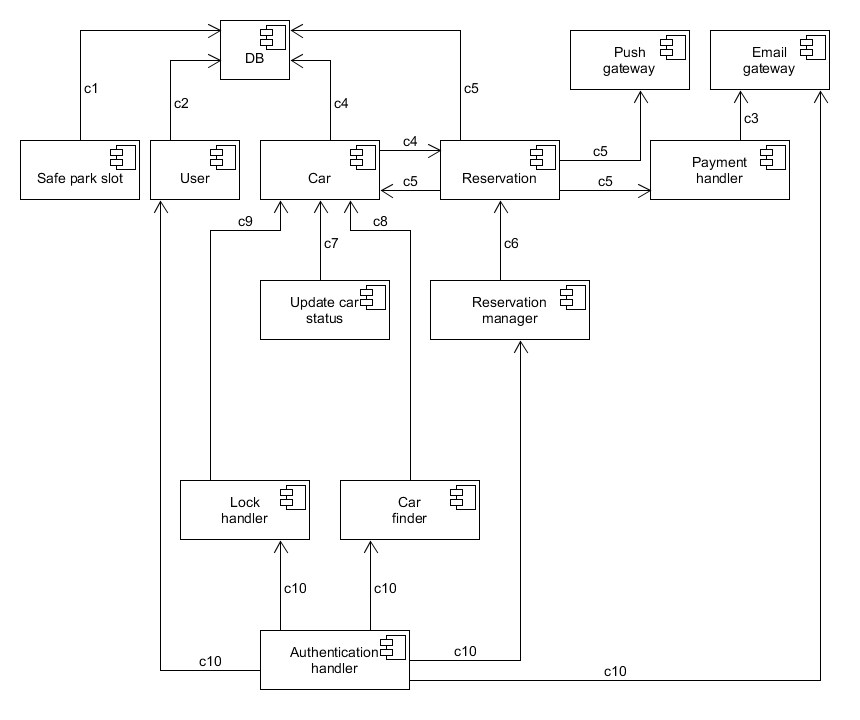
**2.4. Sequence of component/function integration**

**2.4.1. Software integration sequence**

In this section it will be shown the sequence of the components’ integration.

As mentioned above, we consider the high-level components “User”, “Supervisor” and “Physical car” as atomic subsystems, and therefore they are not consider in this section; in the following figure we describe in detail the integration sequence of the components of the “Central server” subsystem.

The bottom-up approach has been respected in the majority of the cases, with only one exception: it’s present a circular dependency between the “Car” and “Reservation” components.

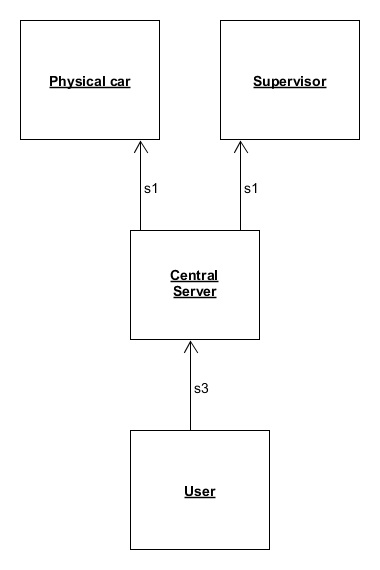


*Central server integration sequence*

|  |  |  |
| --- | --- | --- |
| **N°** | **Component** | **Integrates with** |
| c1 | Safe park slot | DB |
| c2 | User | DB |
| c3 | Payment handler | Email gateway, Bank service(EXT.) |
| c4 | Car | DB, Reservation |
| c5 | Reservation | DB, Car, Push gateway, Payment handler |
| c6 | Reservation manager | Reservation |
| c7 | Update car status | Car |
| c8 | Car finder | Car |
| c9 | Lock handler | Car |
| c10 | Authentication handler | User, Lock handler, Car finder,  Reservation manager, Email gateway |

**2.4.2. Subsystem integration sequence**

For subsystem integration we follow the bottom-up approach too. The reason to do so is that it’s simpler to implement driver for testing the low level components, instead of making coherent stubs of the subsystems. Moreover, a working central server is needed to test in a profitable way the mobile app (User subsystem).

****

*Subsystems integration sequence*

|  |  |  |
| --- | --- | --- |
| **N°** | **Subsystem** | **Integrates with** |
| s1 | Central server | Physical car, Supervisor |
| s2 | User | Central server |