DESCRIZIONE DI COMPONENT:

MAIN SYSTEM COMPONENT view

APPLICATION SERVER

The application server is implemented in the business logic tier using Java EE; it runs on Apache Server.
The access to the DBMS is not implemented with direct SQL queries: instead, it is completely wrapped by the Java Persistence API (JPA).
The business logic is implemented by custom-built stateless Enterprise JavaBeans (EJB).
The application server implements a RESTful API using JAX-RS to allow the clients (web tier and mobile client) to use the services offered by the EJBs.

Api Controller
The API controller replicate many functionalities of the Application Controller, written over. Even there is a duplication of some functionalities, we decided to maintain this controller because we wanted a clear separation between the API and Application.
The Api controller expose the Dbms and other components of the application to external requests. In case of a ride requests (for example) it allow the algorithm to communicate with the DBMS. In case of a mobile use , it makes possible the data exchange from DBMS and application controller.

Application Controller:
The application controller, manage the communication between clients and internal components. It provides the ability to exchange data from an internal component to a client(and so from a client to the system).
This interface is unique and does not depend by other components that are going to use it. It dispatches the message coming from the client to the right internal component.
It manage the ride requests and ride bookings, by call to DBMS and Algorithm.

Algorithm Controller
the main Algorithm controller has to provide a communication between the algorithm pool, and the processes that need one of them. The main algorithm of the system, is the QUI LO DOVETE COMPLETARE VOI.

UserManager
This component manages all the user management features, namely: user login, user registration, user deletion, user profile editing. It also provides a function to confirm the email address provided by the user with the token sent by email, to see and change the user information and the method of payment.

ReservationManager
This component manages all the request of booking. It search if there are available vehicles, it check the trough the helping of the Algorithms the nearest parkings, with available car, it give the possibility to book car. It manage also all the information about the reservation of a vehicle for example the state of the car, where it is placed, how much time the user have to reach the car before that the reservation will expiry, and finally manage also the unlocking of the car.

Utility Manager
This component manages the sending of some notification throughout the email, and also the subtraction of money for the trip.

DATABASE

The database tier runs MySQL Community Edition and uses InnoDB as the database engine: the DBMS has to support transactions and ensure ACID properties. The DBMS will not be internally designed because it is an external component used as a “black box” offering some services: it only needs to be configured and tuned in the implementation phase.
The database can communicate only with the business logic tier using the standard network interface. Security restrictions will be implemented to protect the data from unauthorized access: the database must be physically protected and the communication has to be encrypted.
Access to the data must be granted only to authorized users possessing the right credentials. Every software component that needs to access the DBMS must do so with the minimum level of privilege needed to perform the operations.
All the persistent application data is stored in the database. The conceptual design of the database is illustrated by the E-R diagram SE LO VOLETE METTERE.

USER APP

The user APP implementation depends on the specific platform. The iOS application is implemented in Swift and mainly uses UIKit framework to manage the UI interface. Instead, the Android application is implemented in Java and mainly uses android.view package for graphical management.
The application core is composed by a **Mobile application controller** which translates the inputs from the UI into remote functions calls via RESTful APIs. The controller also manages the interaction with the GPS component using Core-Location framework in iOS app and LocationListener interface in the Android one.

Mobile Application Controller
This is the main controller for the mobile application. it has to show to user the correct view and correct data, in order to make a best fruition of the service in mobile. Via this controller can be made ride request and all the other ride managements in a mobile environment.
it has also the charge of retrive the GPS coordinates, and send them to server.

CAR COMPUTER

The Car computer is saw as a device comparable to a tablet or a smartphone, so with the same architectural structure. It is saw as a thin Client so on it there is a component that is equal with the Mobile Application Controller that we have just seen in the User APP, intact it work base on the same principles.

THIRD PART SYSTEM COMPONENT view

As we can se the two system as the same structure, we use also the same controller-component for the same task And also the same logic. As we know and can see the two system comunicate only throughout the DBMS.
So we will focalize only on the different part:

APPLICATION SERVER

We can see here the two main subsystem:

AdminManager
This component is used from the Admin that access to the system with a Browser and from there can operate very important organization action. So this component manages alla the action of the admin: as bureaucracy thing (fine and assurance expiry), as mantainance and security of the system.

EmployeeManager
This component manage all the task of the employee as the relocation of the car, the plugging of car and all the tusk that he must do too keep the entire carSharing System working.

ADMIN BROWSER (webserver)

The web server is implemented using Java EE web components, namely JavaServer Faces (JSF), which is a server-side framework based on MVC.
The web server is run by Apache Server.
The web tier only implements the presentation layer: all the business logic is handled by the application server tier. The web tier uses the RESTful interface of the application tier.
Using JSF, the view is written as XML files and is completely separated from the logic of the web server. This enables us to write a modular web service.
The web server architecture is composed simply by JSF
and by a controller class that takes the admin input and translates it in API requests.