PowerEnJoy

A new way of doing car sharing

March 10th, 2017

Luca Marzi
Valeria Mazzola
Federico Nigro

Topics

- Problem analysis
- Use Case analysis
- Goals and Requirements
- Overview of the system’s runtime behavior
- Architecture
- Testing strategy
- Cost estimation and planning
- Risk evaluation

Problem analysis

We manage to design and develop a digital management system for a car-sharing service that **exclusively employs electric cars**

Why electric cars only?

Because big cities are already experiencing high levels of pollution, and their citizens are looking for a new way to move that may be, at the same time, **cheaper and green**.

Problem analysis

- The system will allow registered users to **discover available cars nearby** their current position
- Registered users will be allowed to **book a car for a limited time** (1 hour)
  - Within this time, the user will be allowed to **delete his/her reservation**
- The user will be asked to pay a fee of 1 euro if the reservation expires
- After a reservation is concluded or expired, the reserved will become available for other users
- The system will incentivize the **virtuous behaviors of the users** by means of applying discounts or fees in the appropriate contexts.

Status of the Market

Only in the city of Milan there are approximately:

- **6** car sharing services involving only vehicles with **combustion engines**
- **3** car sharing services involving only **electric vehicles**

Use case analysis

Main use cases analysis

In this section we aim to provide the description those use cases that better capture the context of the system to be and the main functionalities it will have to provide, such as:

- **FindAvailAbleCar**: the case in which the user looks for a car to book in the nearby
- **ReserveAvailableCar**: the case in which the user reserves and available car
- **StartRide**: the user converts a reservation in a ride
- **StartActiveRide**: the user starts the car’s engine and uses the car
- **EndActiveRide**: the user stops the car’s engine
- **EndRide**: the user leaves the reserved car
- **EndActiveReservation**: the user decides to close his reservation and let the car be available again to other users

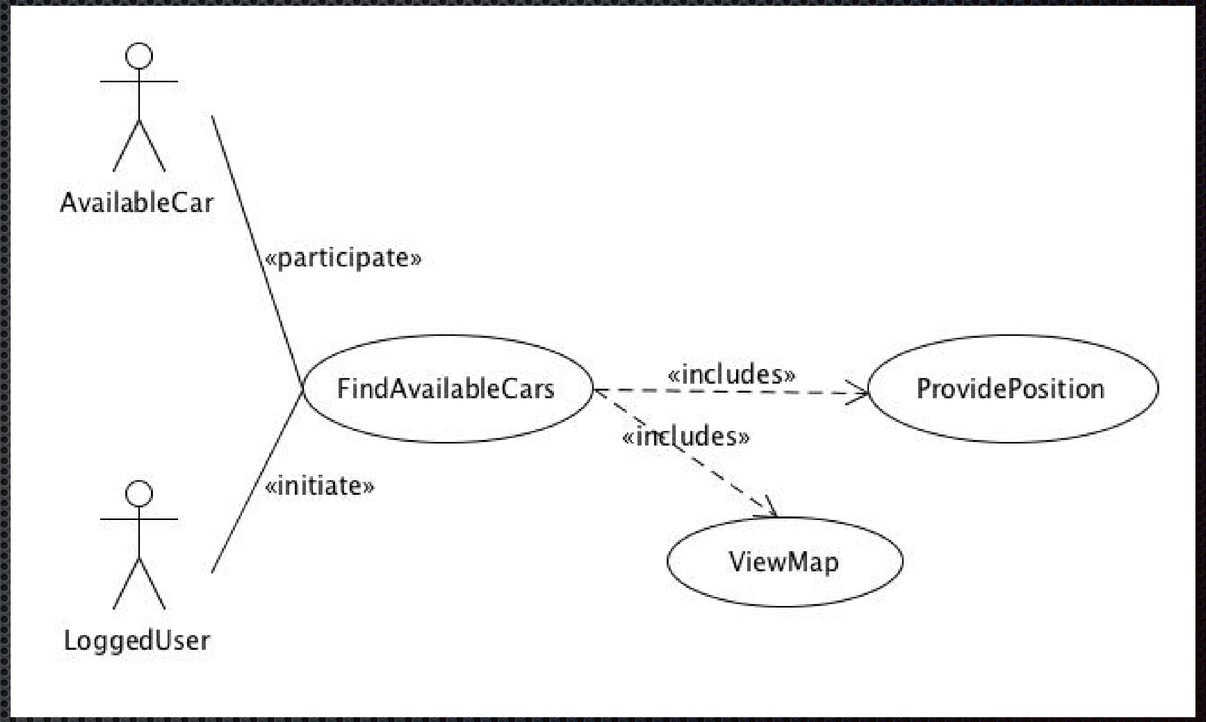
FindAvailableCar

Actors

- LoggedUser
- AvailableCar

Entry Condition

- The user communicates the will to find available cars nearby



FindAvailableCar

Flow of events

- The LoggedUser submits a valid position to the System
- The System searches AvailableCars within a certain distance from the position submitted by the User

Exit Condition

- The System shows to the LoggedUser a map showing the position of AvailableCars that have been found

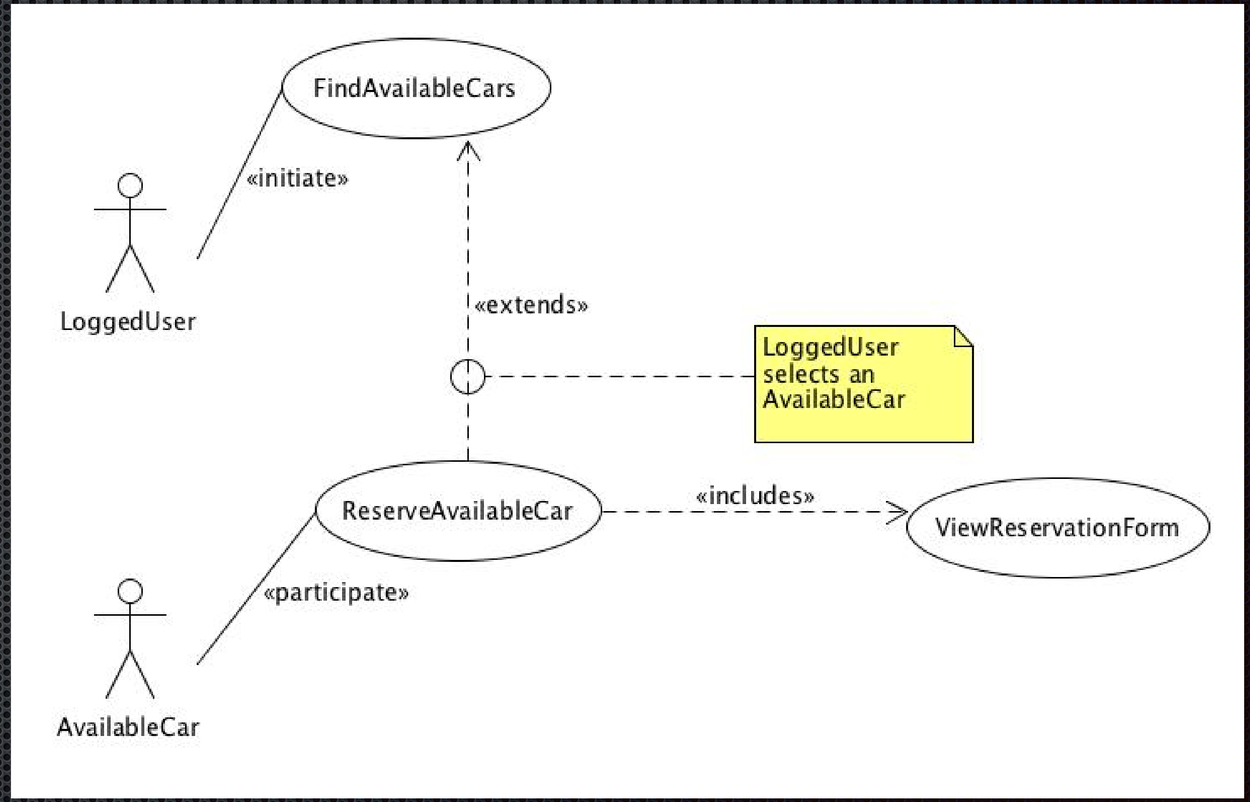
ReserveAvailableCar

Actors

- LoggedUser
- AvailableCar

Entry Condition

- LoggedUser selects an AvailableCar from the ones that are on the map that shows the AvailableCars nearby the LoggedUser's position



ReserveAvailableCar

Flow of events

- The System shows the reservation form for the specific car whose identity is provided by the LoggedUser
- The LoggedUser sees all the information about the AvailableCar provided by the reservation form
- The LoggedUser confirms the will to reserve the AvailableCar via the Network

Exit Condition

- The System confirms the reservation for the AvailableCar and shows to the LoggedUser the information about the active reservation that has been placed

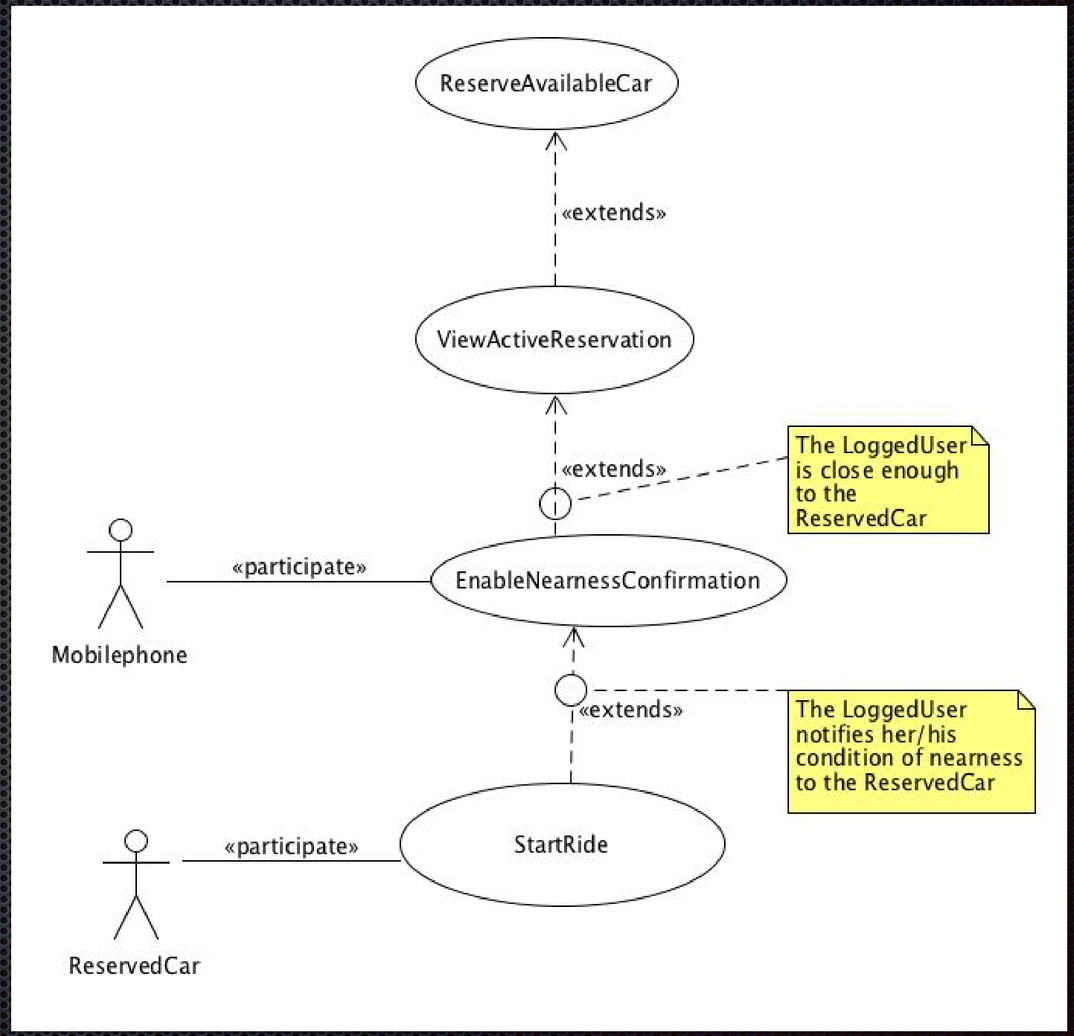
StartRide

Actors

- LoggedUser
- ReservedCar

Entry Condition

- The LoggedUser views her/his active reservation for the ReservedCar and the GPS position of the LoggedUser is within a certain distance from the ReservedCar



StartRide

Flow of events

- The System sends to the LoggedUser a notification about the possibility to notify her/his condition of nearness to the ReservedCar
- The LoggedUser submits to the System her/his condition of nearness to the ReservedCar via the Network.
- The System unlocks the ReservedCar
- The System associates a ride to the active reservation
- The System starts charging the LoggedUser for the the parking fee
- The System informs the LoggedUser about the beginning of the ride and the state of the charging via the on board system of the ReservedCar

StartRide

Exit Condition

- The System notifies the beginning of the ride to the LoggedUser

Special requirements

- The ReservedCar must be unlocked by the System within 1 minute from when the LoggedUser notified the condition of nearness to the ReservedCar
- The association the ride to the the active reservation shall be transactional.

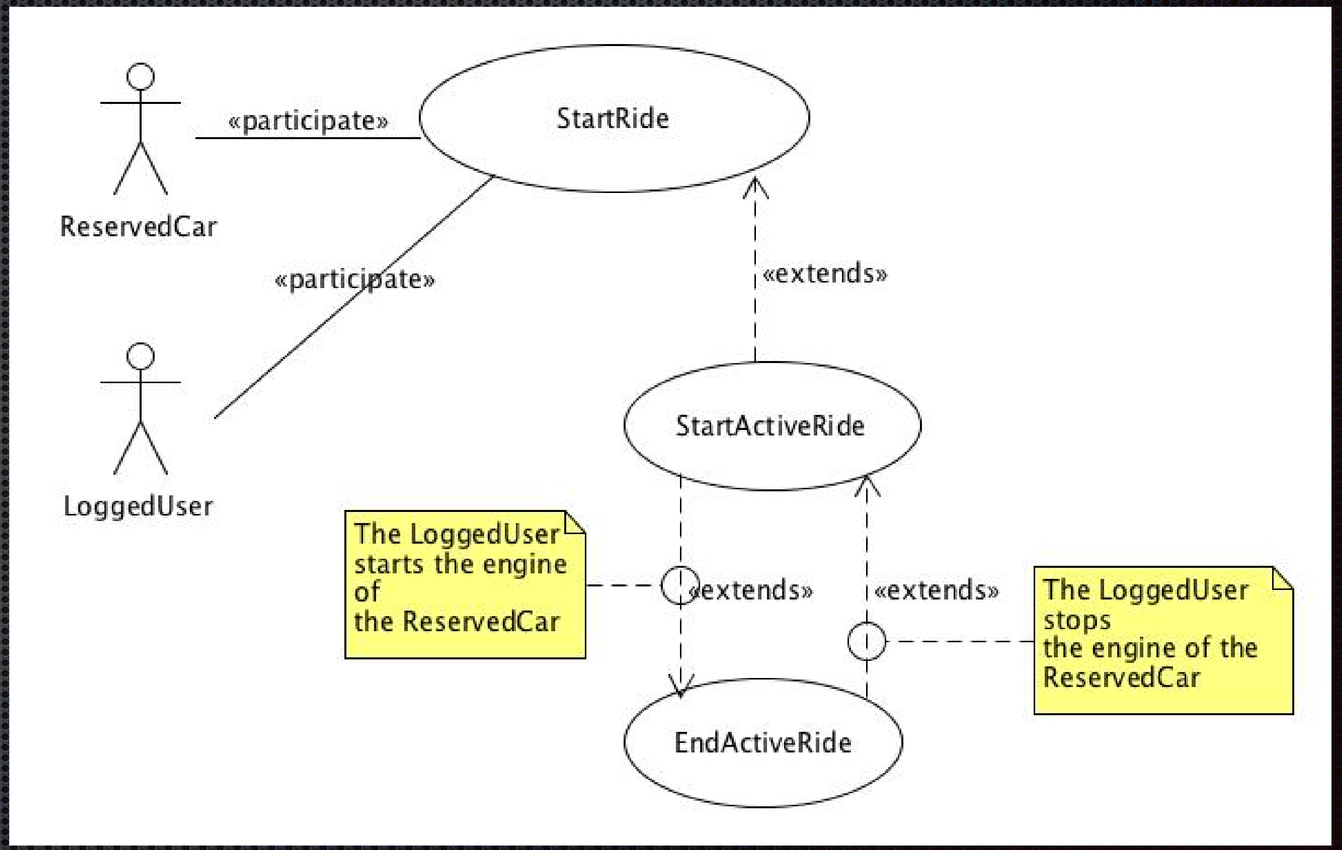
StartActiveRide

Actors

- LoggedUser
- ReservedCar

Entry Condition

- The LoggedUser starts the engine of the ReservedCar



StartActiveRide

Flow of events

- The System associates a new active ride to the ride
- The System starts charging the LoggedUser for the active ride's tariff
- On the basis of the state of the ReservedCar the System applies discounts and penalties to the active reservation of the LoggedUser.
- The System informs the LoggedUser about the beginning of the active ride and the state of the charging via the Car's System of the ReservedCar

Exit Condition

- The System notifies the beginning of the active ride to the LoggedUser

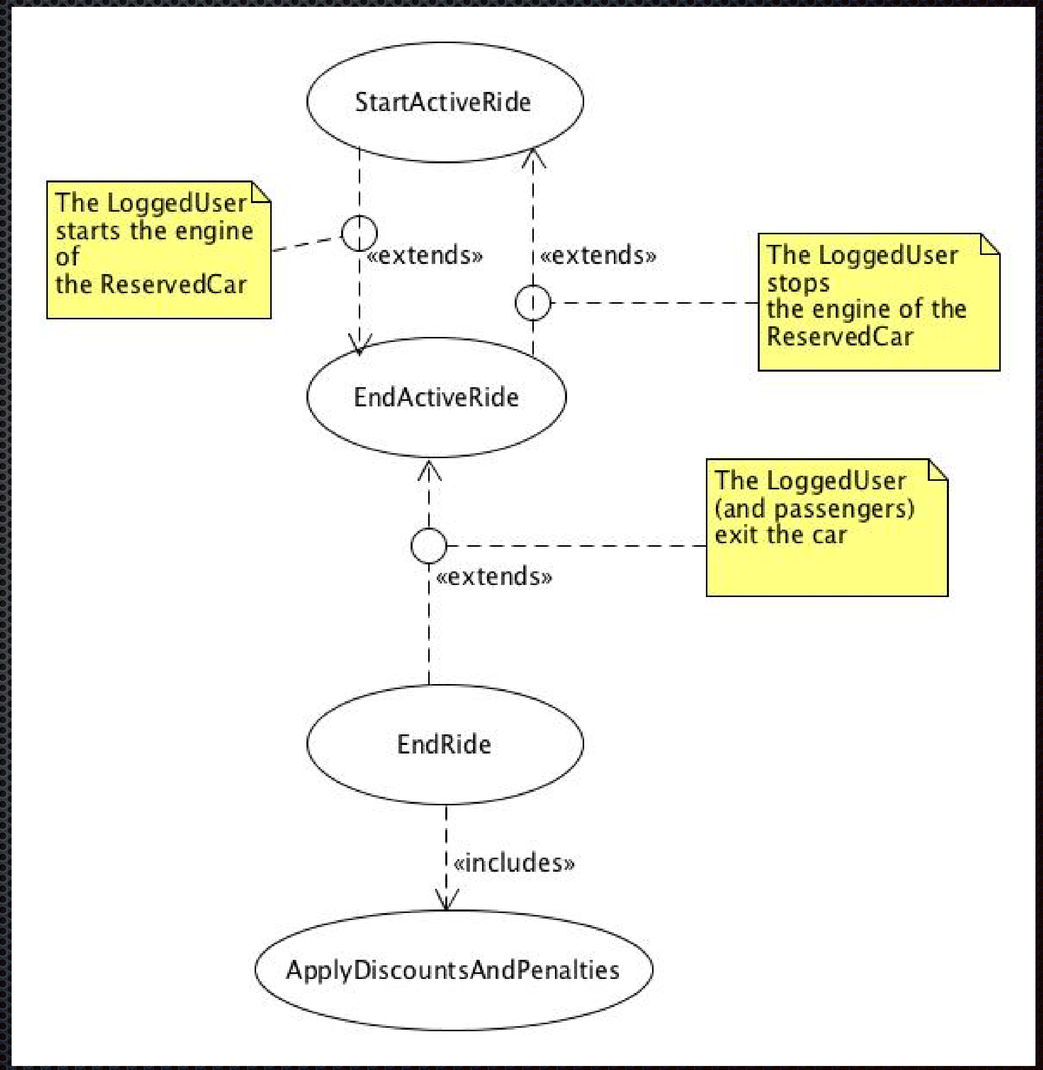
EndActiveRide

Actors

- LoggedUser
- ReservedCar

Entry Condition

- The LoggedUser stops the engine of the ReservedCar



EndActiveRide

Flow of events

- The System ends the active ride.
- The System stops charging the LoggedUser with the active ride's tariff
- The System starts charging the LoggedUser for the the parking fee
- The System informs the LoggedUser about the ending of the active ride and the state of the charging via the Car's System of the ReservedCar

Exit Condition

- The System notifies the ending of the active ride to the LoggedUser on the Car's System of the ReservedCar

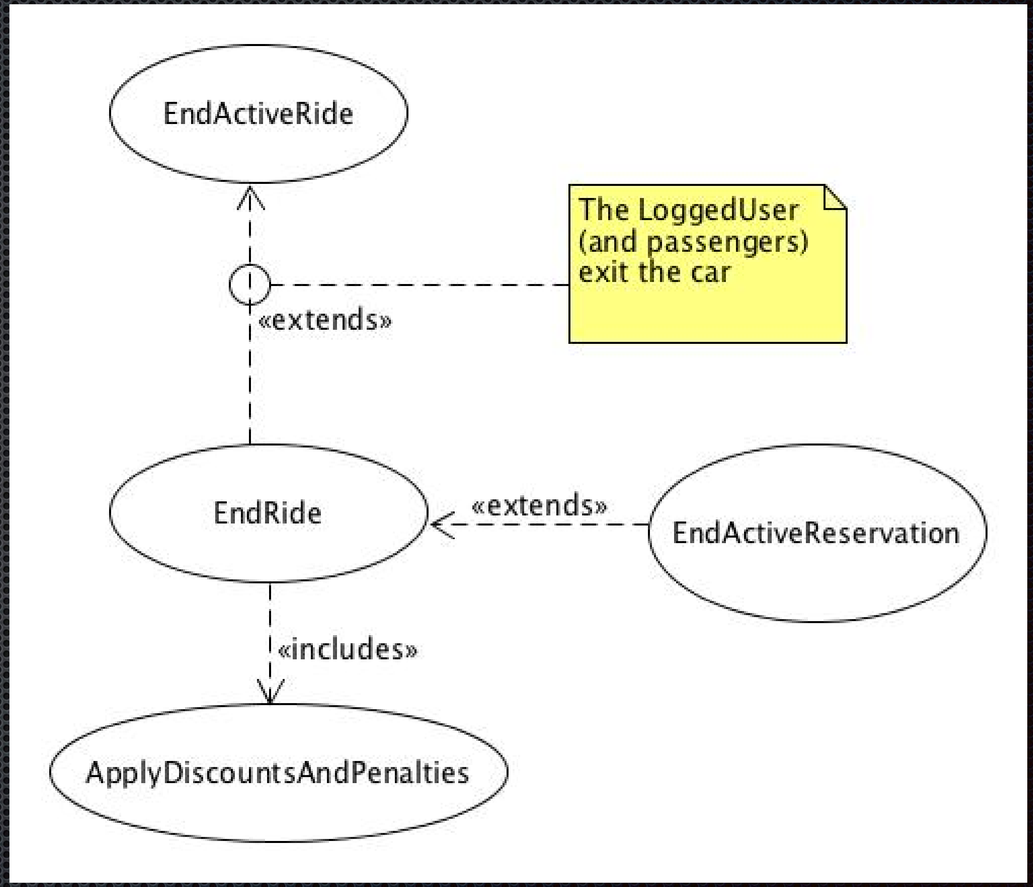
EndRide

Actors

- LoggedUser
- ReservedCar

Entry Condition

- The LoggedUser and other passengers exit the ReservedCar



EndRide

Flow of events

- The System ends the ride associated to the active reservation of the LoggedUser for the ReservedCar
- On the basis of the state of the ReservedCar, the System associates discounts and penalties to the active reservation
- The System locks te ReservedCar
- The System stops charging the LoggedUser for the parking ride's tariff
- The System informs the LoggedUser about the ending of the ride and the state of the charging via a notification

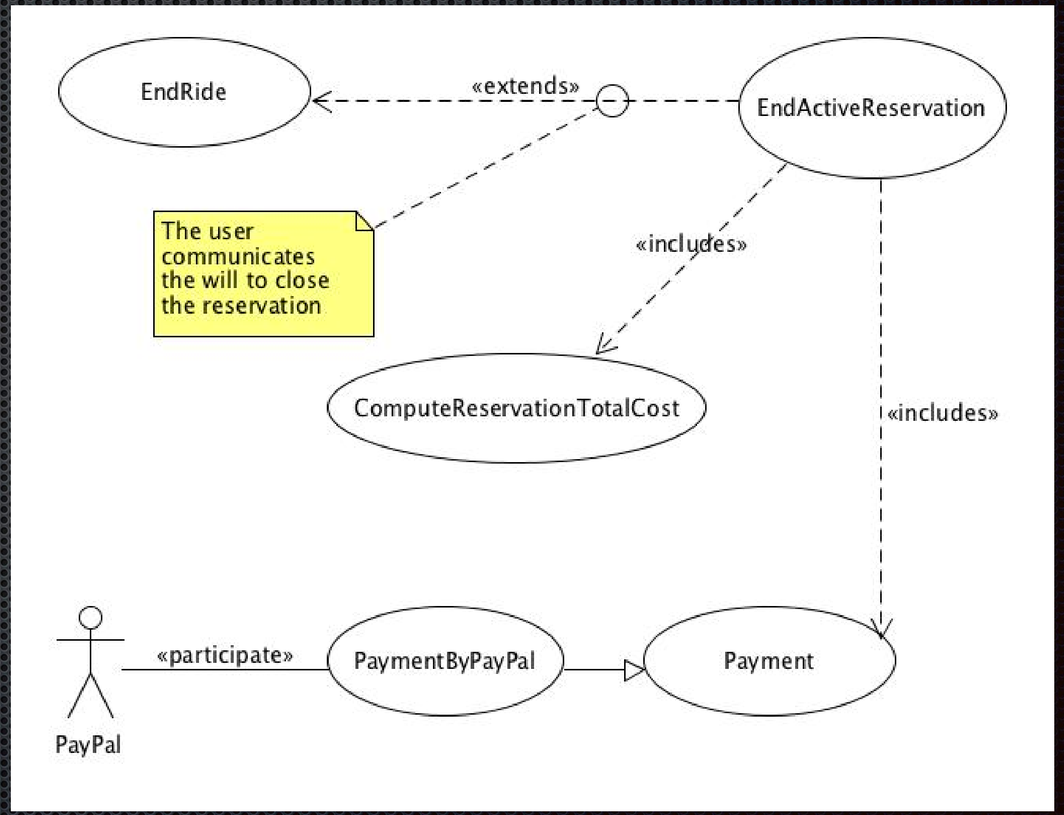
Exit Condition

- The System notifies the ending of the ride to the LoggedUser

EndActiveReservation

Actors

- LoggedUser
- ReservedCar



Entry Condition

- The user concludes a Ride
- The LoggedUser communicates the will to conclude his/her reservation

EndActiveReservation

Flow of events

- The System ends the active reservation of the LoggedUser for the ReservedCar
- The System computes the total cost of the active reservation and charges the LoggedUser for that amount of money
- The System executes the payment for the total cost of the active reservation
- The System sends a receipt of the ended active reservation and the payment to the LoggedUser via email
- The System informs the LoggedUser about the ending of the active reservation via a notification

Exit Condition

- The System notifies the ending of the ride to the LoggedUser

Goals and Requirements

Goals and functional requirements

Summary

Goals	Corresponding Requirement	RASD corresponding diagram(s)	DD corresponding diagram(s)
[G1.1] (Registration topics) [G6.1] (Errors alerting topics)	• **[R1.1]:** processing the registration and controlling inputs • **[R1.2]:** alerting the User in case of error • **[R1.3]:** storing data • **[R6.1]:** errors alerting and restore point	**(13.1) Registration Phase** *Interactions only between User and System in a general way satisfying **[R1.1]** and **[R1.2]**. **[R6.1]** is satisfied by the mean of cycles in the diagram, which makes implicitly the System to understand when "restart" the operation towards the User.*	**(2.8.1) Registration** *The integration of the **Mobile Application** and the **Database System** are also considered, in addition to the RASD document diagram **[R1.3]** is satisfied too.*

Goals	Corresponding Requirement	RASD corresponding diagram(s)	DD corresponding diagram(s)
[G1.1] (Registration topics) **[G6.1] (Errors alerting topics)**	- **[R1.1]:** processing the registration and controlling inputs - **[R1.2]:** alerting the User in case of error - **[R6.1]:** errors alerting and restore point	**(13.2) Authentication Phase**	- **(2.8.2) Authentication**
[G2.1] (discovering positions of cars) **[G2.2] (updating cars positions)** **[G2.3] (multiple requests management)** **[G3.1] (cars information and reservation form)** **[G3.2] (reservation correctness and completion)** **[G6.1]** **[G7.1] (mailing service)**	- **[R2.1], [R2.2], [R2.3], [R2.4]:** internal functions and instances management - **[R3.1]:** instances wrt the Database - **[R3.2]:** instances correctness - **[R3.3]:** user's request confirmation - **[R7.1]:** automatic messages management.	**(13.3) Searching and Booking a nearby car** ***[R2.1], [R2.2], [R2.3], [R2.4]*** *are correctly handled considering that the System (the unique entities in addition to the User in this case) implicitly makes the needed actions for guaranteed them.* *The presence of cycles and recurrences in the diagrams points out that the* ***[R3.2]*** *and* ***[R3.3]*** *are managed as well*	**(2.8.5) Searching and Booking a nearby car** *In addition to what has been said for the RASD diagram, we can confirm the respect of the requirements* ***[R3.1]****, because the mobile app and the Database are integrated in this new Diagram and their interactions too.* *Concerning the* ***[R7.1]*** *the assumption is that his management is implicit: when a ride is confirmed and taken in the account of the System, a mailing service advice the User of those events as guarantee.*

Goals	Corresponding Requirement	RASD corresponding diagram(s)	DD corresponding diagram(s)
[G4.1] **(reason of expiration)** **[G4.2]** **(consequences towards the User)** **[G4.3]** **(User's possible actions within the our)** **[G6.1]**	▪ **[R4.1]:** the System keeps track of the elapsed time ▪ **[R4.3]:** reservation is tagged as expired by the System **[R4.5]:** the System advises the User if the reservation is expired **[R6.1]:** errors alerting	**(13.4) Time expiration** *All the requirements expressed in the previous column were showed with this Diagram: the System sets a countdown from the moment that the reservation was made. The User is notified if the associated time expired.*	(2.8.6) Time expiration *In this case the System alerts the User about the expiration by the mean of the Mobile App, the concepts are identical.* ***Note: there are some extra requirements which are not exposed by the diagrams: they will be discussed later.***
[G5.1], [G5.2], [G5.3] **(condition of nearness, when the car shall be unlocked)**	▪ **[R5.1], [R5.2], [R5.3]:** actions taken by the System wrt the definition of the corresponding numbered goals.	***(13.5) The System unlocks a Car*** *There's a precise schema of actions for the Diagram to be correct through the definitions of the goals and requirements.* *The basic interactions between the User, the System and the System and the On Board System are shown for the case.*	*(2.8.8) Unlocking a Car* *Some analogies are shared between this diagram and the previous one. In this diagram is also introduced the Mobile App, but the chains of events is equal.*

Goals	Corresponding Requirement	RASD corresponding diagram(s)	DD corresponding diagram(s)
[G5.6] **(end of ride conditions)** **[G5.7]** **(penalties conditions)** **[G5.9], [G5.10], [G5.11], [G5.12], [G5.15], [G5.16]** **(discounts and fees conditions)**	• **[R5.6], [R5.7], [R5.8], R[5.9], [R5.10], [R5.11]:** the requirements listed refer to the various condition of extra payment or a discount towards the User • **[R5.12]:** refers to the interaction between the System and the Transaction System. This last entity alerts the System when a transaction was well made or not.	**(13.6) End of the ride and payment** *The Diagram in this case is almost completed. It aims to show the interactions in three steps (User-System, User-Payment System, Payment System-System). Requirements are satisfied when the interactions between the Car's System and the System take place and when the interactions between System and Payment System are correctly made.*	**(2.8.4) Finish a ride** *At this level, the exchanging of information between System and Mobile App is also pointed out. When the User finish the ride, it continues the operation of payment from its App and this is exactly the concept added by this diagram.*

Goals	Corresponding Requirement	RASD corresponding diagram(s)	DD corresponding diagram(s)
[G5.13] (money saving option conditions) [G5.14] (seeing the list of parking area) [G5.15] (activating the discount) [G5.16] (total cost computation)	• **[R5.13]:** System actions towards the User decision to activate the money saving option • **[R5.14]:** computing the list of the safe parking area and showing it to the User • **[R5.15]:** final cost computation (taking into account discounts and fees to be applied).	**(13.7) Activating the “Money Saving Option”** *The Diagram show all the steps followed by the User, the Car's System and the central System for obtaining the path to the nearest safe parking area as a result on the Car System display.*	**(2.8.3) Activating the “Money Saving Option”** *In this case the Diagram is equal to the rasd one. The various computation of the central System and the Car's one brings at same conclusions.*
[G5.8] (car locking conditions)	**[R5.8]:** actions of the System wrt the respective goal	*The Diagram is not present in the document.*	(2.8.7) Locking a Car
		The Diagram is not present in the document.	(2.8.9) Telemetry

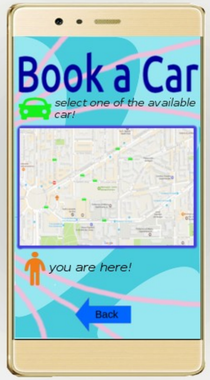
Non functional requirements

Simple mockups

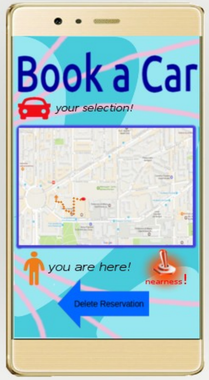
1)

2)

3)



4)



Sequence diagram analysis

Entry conditions to the analysis

Before starting to show the Diagrams in details, it is important to bear in mind some preliminary considerations about the various interactions among the User, the Application and the Environment in which the two entities are centered:

- Software pre-requisites are well set
- There isn't any problems of internet connection, GPS services in their software components and in their infrastructures
- In general: everything concerning external elements of environment which supports the role of application, works fine for the purpose.

Sequence diagram analysis

First analysis

- A first analysis of the Sequence Diagrams brought to a “glimpse” of the interactions between the various Actors in the entire System

- These diagrams can be found in the RASD document and take into account the first definition of goals and requirements to be satisfied, as well as the actors and the scenarios

Sequence diagram analysis

Second analysis

- The detailed analysis of the Diagrams shows some important features came up during the design of the Architecture

- In these diagrams there is a deeper spectrum of the Actors and Parts involved in the interactions

- A first draft of the implementation is schematized too, the objective of the Diagrams are not to go through the aspect, but to surely give some important guidelines for the complete project of the topic.

Sequence diagram analysis

Catalog

RASD Document

- (13.1) Registration Phase
- (13.2) Authentication Phase
- (13.3) Searching and Booking a nearby car
- (13.4) Time expiration
- (13.5) The System unlocks a Car
- (13.6) End of the ride and payment
- (13.7) Activating the “Money Saving Option”

Design Document

- (2.8.1) Registration
- (2.8.2) Authentication
- (2.8.5) Searching and Booking a nearby car
- (2.8.6) Time expiration
- (2.8.8) Unlocking a Car
- (2.8.4) Finish a ride
- (2.8.3) Activating the “Money Saving Option”

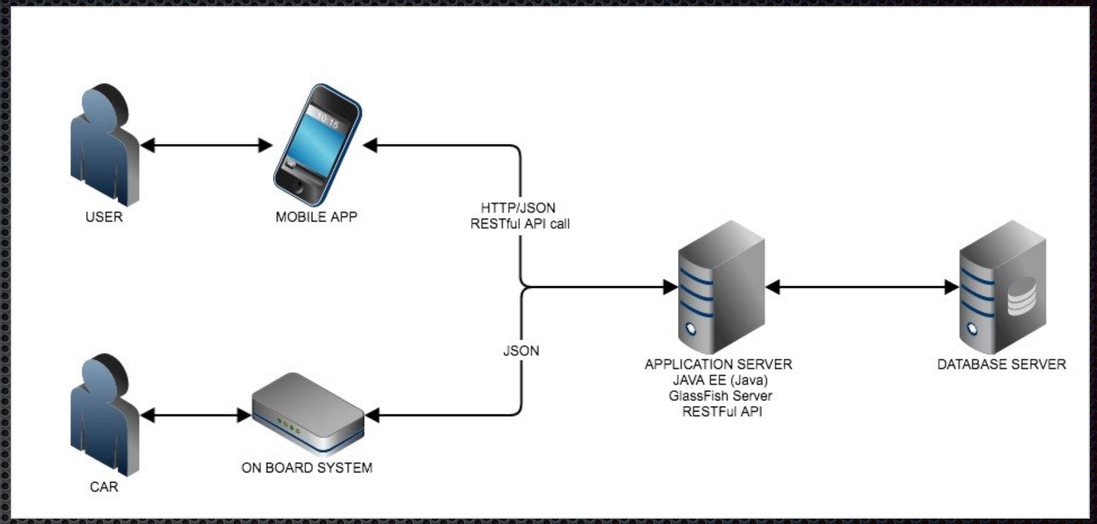
- (2.8.7) Locking a Car
- (2.8.9) Telemetry

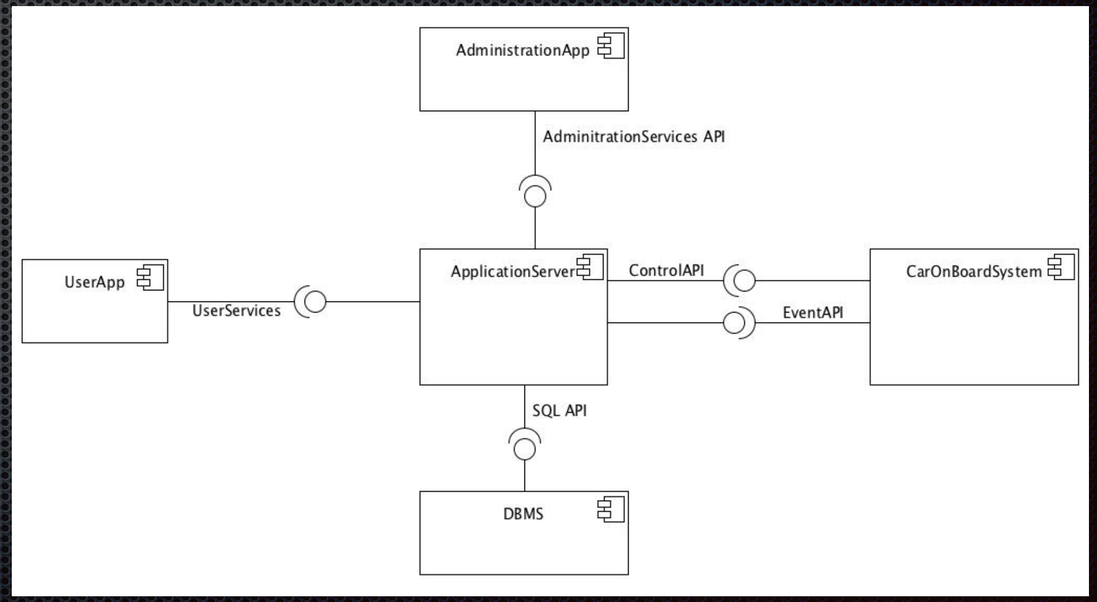
Architecture

Architecture

High level overview

- Mobile application: software component installed on the user’s device which renders the user interface and handles interactions with the user
- Administration application: web application meant to provide all those functionalities required for the administration of the system
- Car on board system: set of software components installed on the car’s system (provided by the manufacturer which handles car’s relevant sensors and commands coming from the application server
- Application server: set of software components that handle the business logic of the system
- Database server: set of software components which allows to store permanent data.





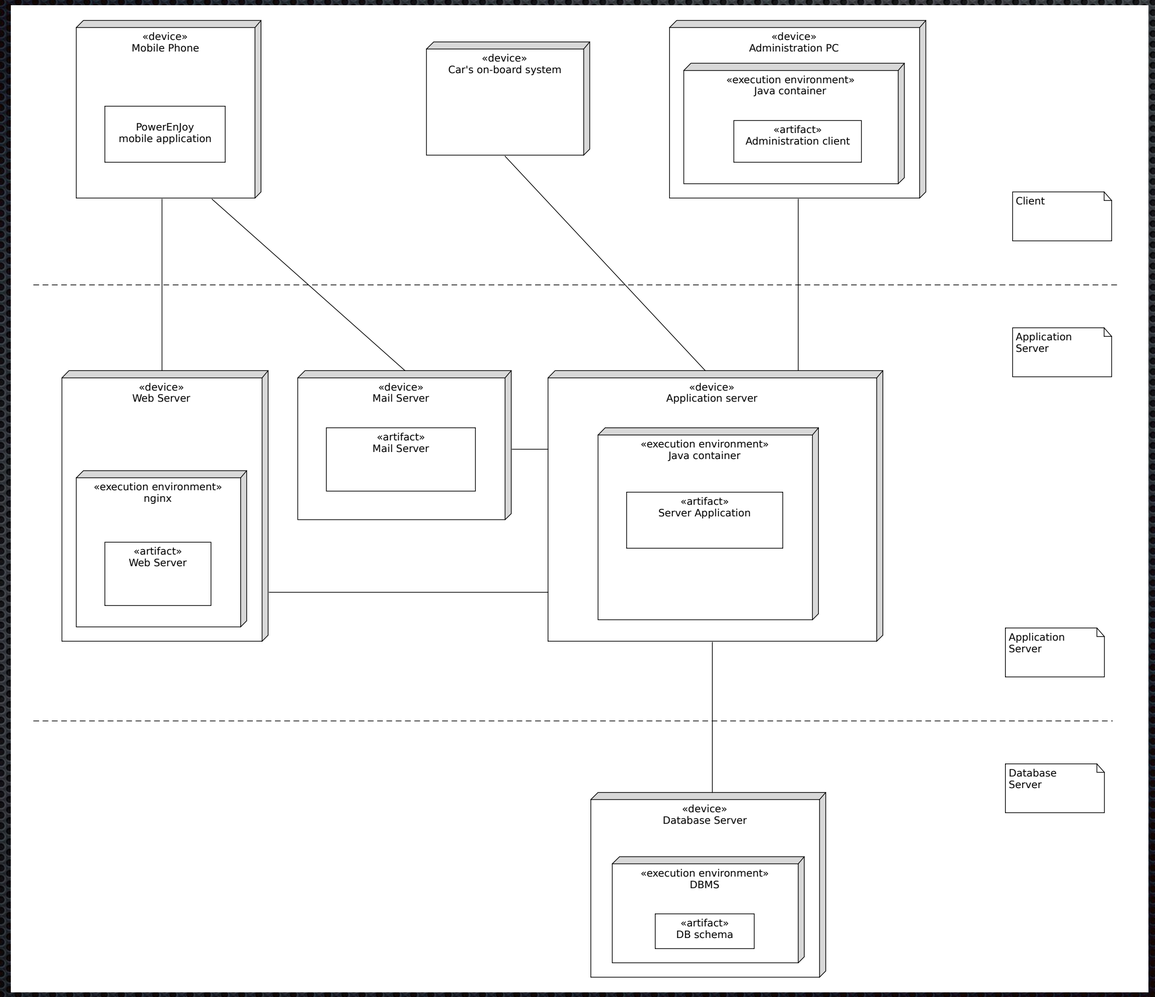
Architecture

- The PowerEnJoy system is designed with a modified MVC approach:
- The Model is in the Business Manager
- Multiple Controllers manage multiple aspects of the system
  - User Controller
  - Car Controller
  - Payment Manager
  - Administration Helper

Architecture

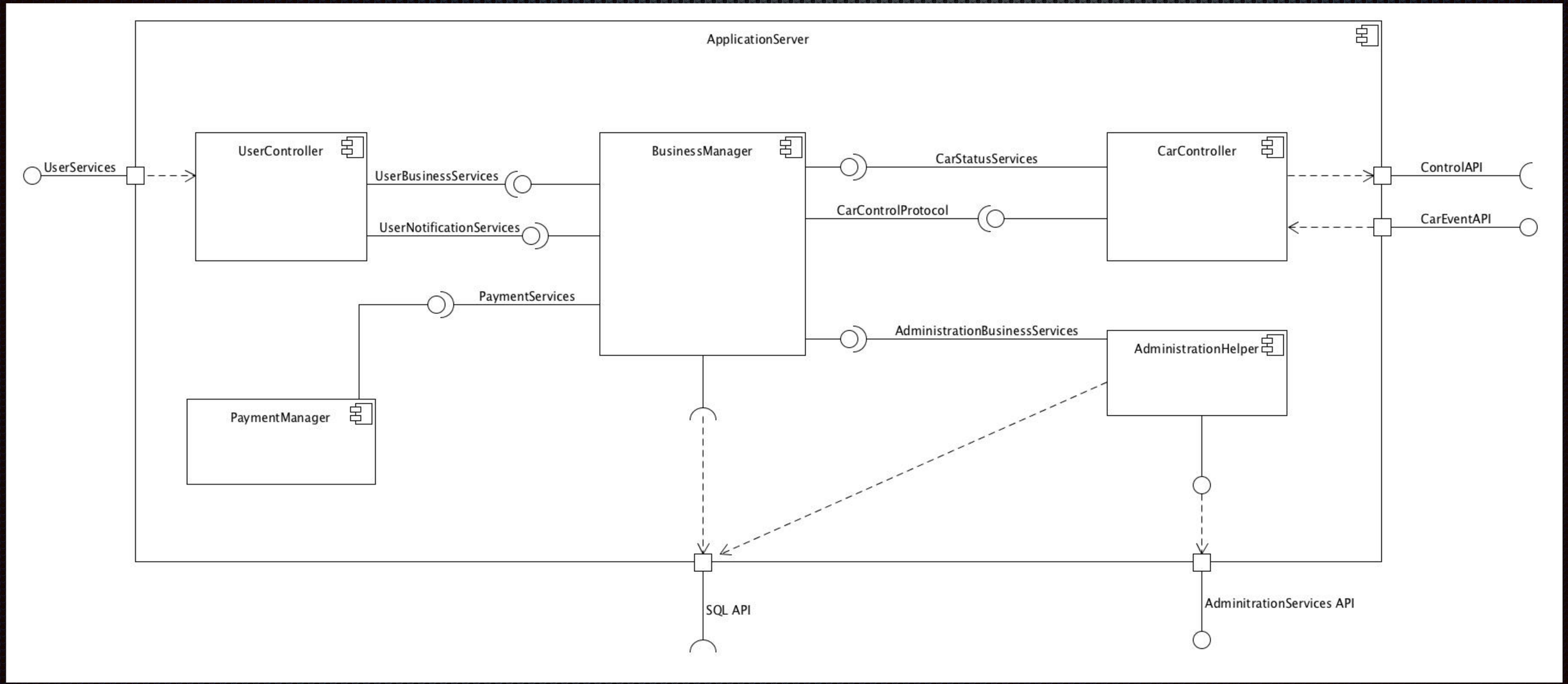
- The PowerEnJoy system is designed with a modified MVC approach:
- The Customer View is implemented:
  - in the Mobile App
  - on the car system
- The Administration View is offered by the server as a web application (thin client approach)

Architecture: deployment



Application server

High level description



Application server

High level description

- Exposes a RESTful API to the Mobile application to provide all the functionalities that a user might call directly or indirectly
- Exposes a RESTful API to the Administration Web application to provide all the functionalities that the administration of the service needs.
- Exposes a RESTful API to the Car on board system to allow it to notify events of interests from the the point of view of the business logic (i.e. car engine starts)
- Forwards commands to the Cars by means of the software API provided by the Car on board system
- Interacts directly with the Database server by means of the standard SQL interface (2011)

Application server

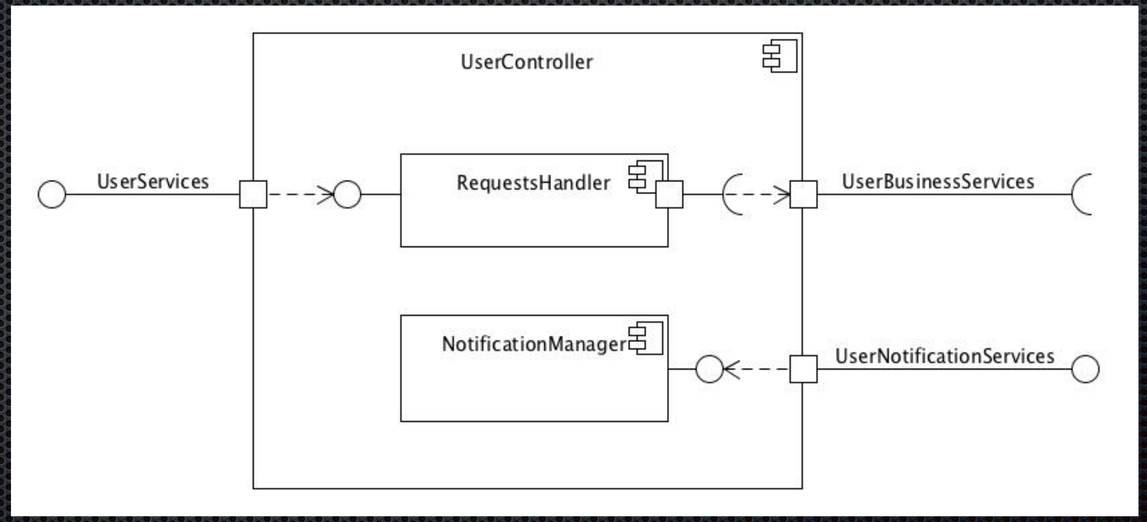
Components’ description

- Business Manager: software component which handles the entire business logic of the system
- User Controller: handles the bidirectional communication between the mobile applications and the application server.
- Payment Manager: software component which handles all the interactions between the business logic and with third-part payment services (by now, PayPal is the only one supported)
- AdministrationHelper: software components which handles all the requests coming from the administration’s web application (AdministrationServices API)
- CarController: handles the bidirectional communication between the on board car system and the application server.

User Controller

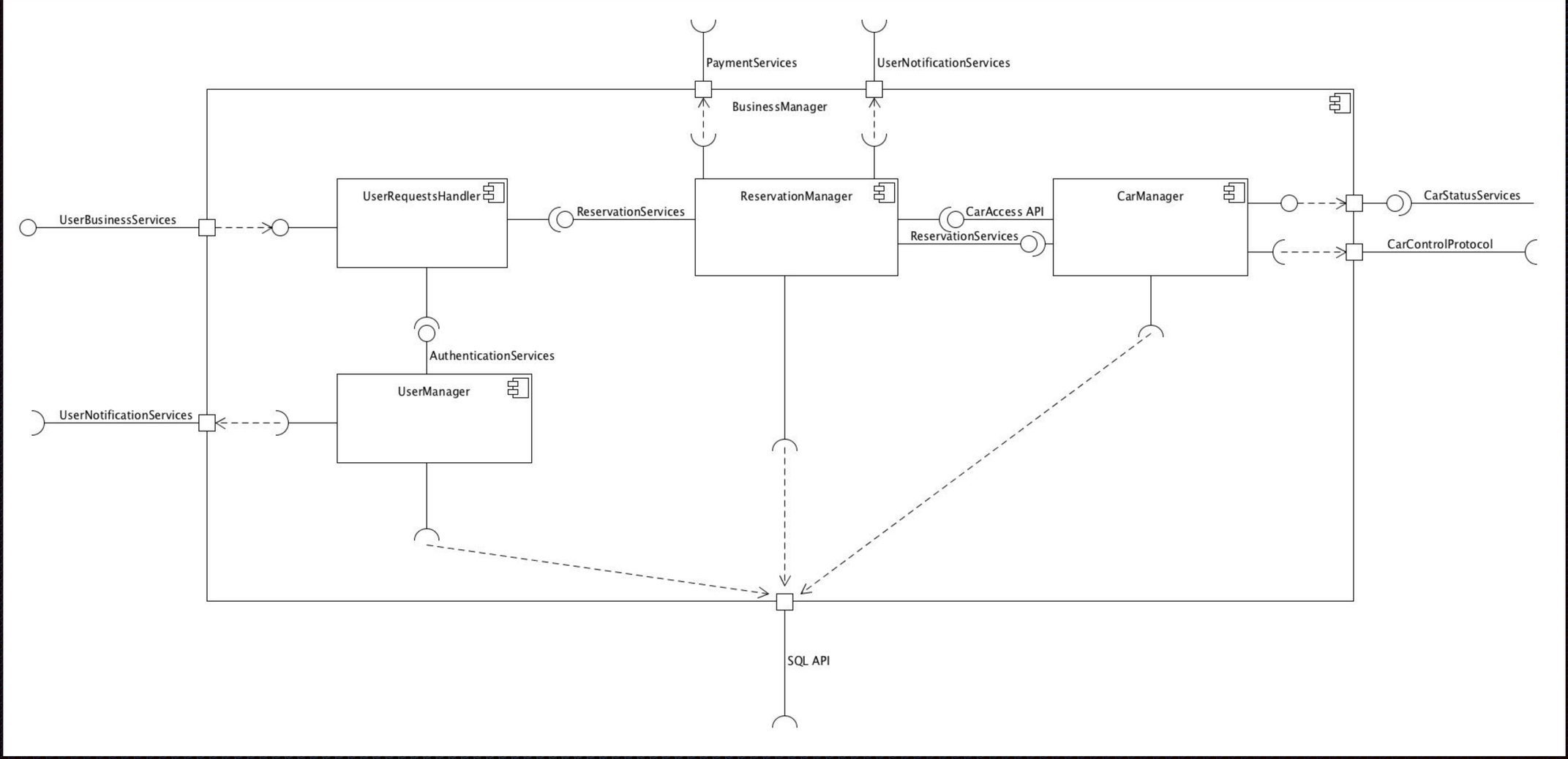
High level description

- Following the MVC paradigm, this controller performs checks on the incoming user requests
- It also exposes an interface to provide means to forward notifications of various type to the user.



Business Manager

High level description



Business Manager

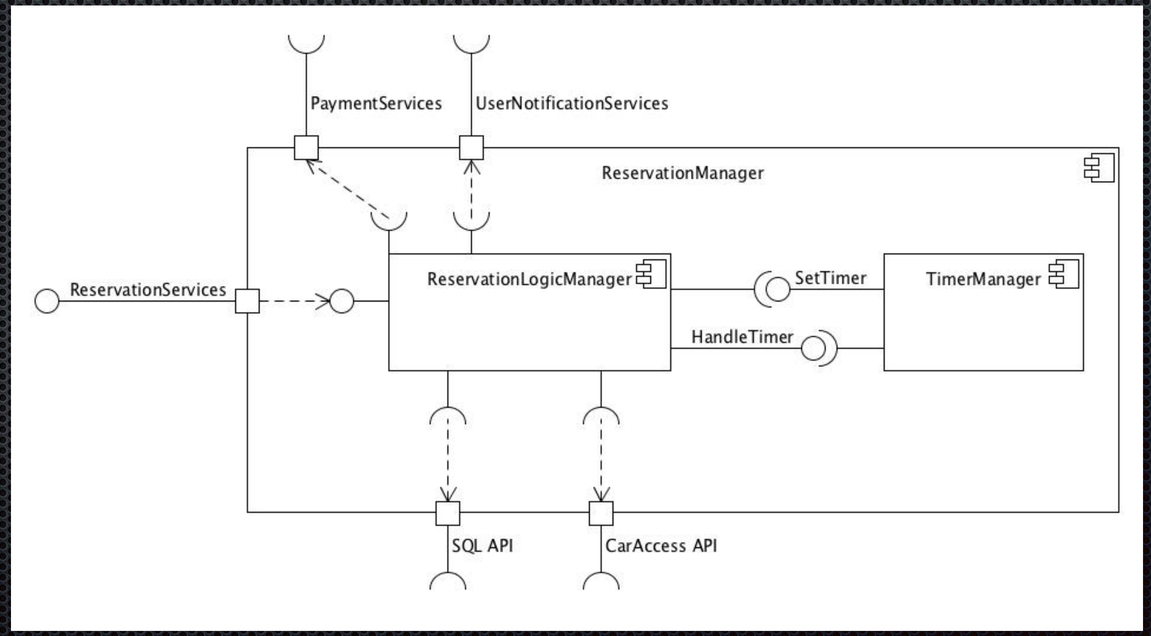
High level description

- Exposes interfaces to provide all the business logic functionalities both to users and physical car.
- Relies directly on the SQL interface to interact with the database services
- Relies on the UserNotificationServices interface provided by the UserController to forward notifications to the users.
- Relies on the CarControlProtocol interface provided by the CarController to forward commands to the cars.
- Relies on PaymentServices provided by the Payment software component

Reservation Manager

High level description

- It Is one of the core software components of the business logic
- The Reservation Manager is in charge of all reservation operations, including the computation of a reservation's cost.



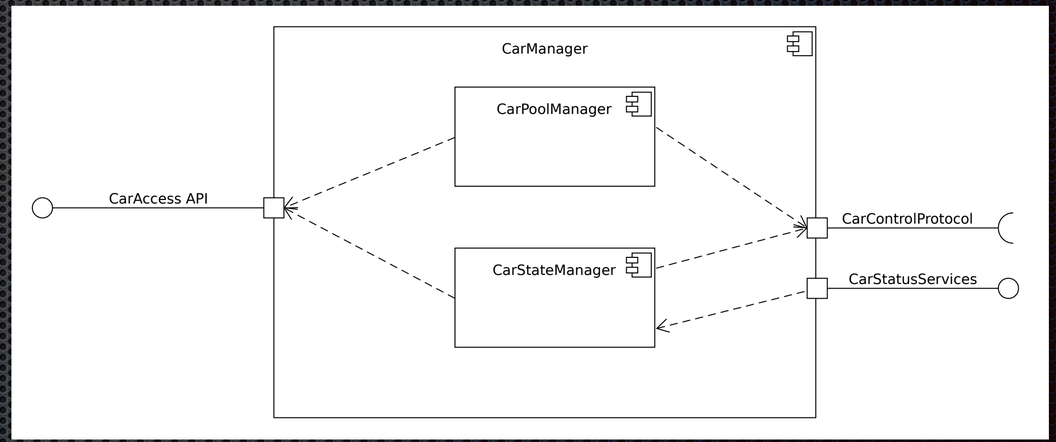
Architecture: the car subsystem

The car subsystem consists of:

- The Car Manager: part of the Business Manager (Model)
- The Car Controller: handles communication with the cars
- The Cars themselves as their cyber-physical systems

Architecture: the Car Manager

- Part of the Business Manager
- Performs all DB operations regarding:
  - Car state handling (Car State Manager)
  - Subscription or deletion of cars to the system (Car Pool Manager)
- Answer queries on car sets issued by the user

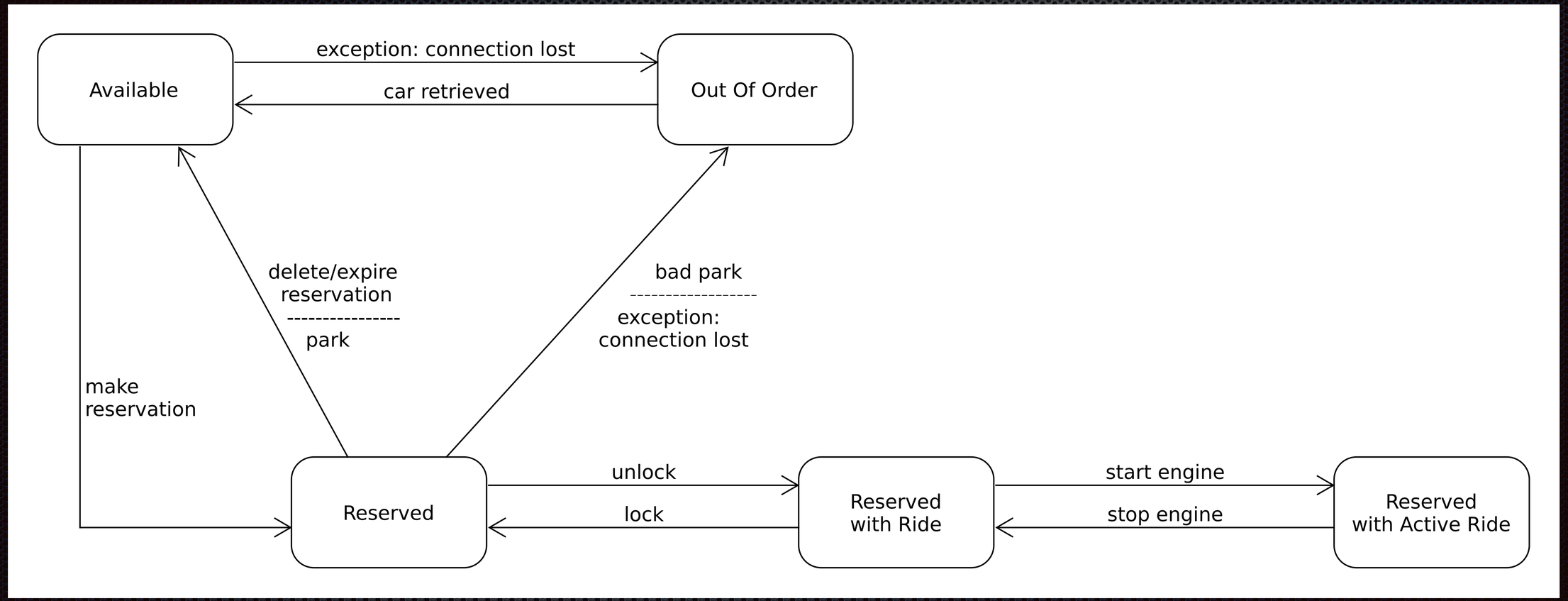


The Car communication protocol

Cars will interact with the server’s Car Controller in the following ways:

- Authentication to the server
- Ordinary communication
  - **Events**: state changes originating from the car
  - **Commands**: state changes issued to the car by the server
  - **Ordinary telemetry**: regular messages to update the server about the car general condition (position, battery status...)

The Car as a FSM



The Car as a FSM

State changes issued by the server (commands):

- **Reserve**: an available car becomes reserved
- **Free**: a reserved car becomes available
- **Invalidate**: a reserved car is parked in an unreachable spot and becomes Out Of Order
- **Unlock**: the user has pressed the unlock button on their phone, the reserved car now is Reserved With Ride
- **Retrieve**: an Out Of Order car has been retrieved by personnel and becomes available again

The Car as a FSM

State changes originating from the car (events):

- **Lock**: the user has closed and locked the car door after a ride, the car state changes from Reserved With Ride to Reserved
- **Start engine**: the car state changes from Reserved With Ride to Reserved With Active Ride
- **Stop engine**: the car state changes from Reserved With Active Ride to Reserved With Ride

Testing

Testing

Entry criteria

- interactions among entities presented in the Architecture and Sequence Diagrams are in their final version
- functionalities are well defined too
- there is already a decision on which tools will be used for the various implementations

Testing

Strategy: bottom-up

- starting to the lowest level of abstraction and integrating the components of the System for reaching the highest one (final tests on the entire system in its final state)
- the Top down strategy could be useful too in some test case, especially when there is the testing of some User functionalities, such as the GUI utilization one

Testing

Bottom-up decisions

Here there are the description of the steps followed for the test and integration of the components of the System:

- the sub components concerning the functionalities of the System towards the User, the Management Unit Subsystem, the Payment System, the Database Subsystem are defined
- in this phase there could be designed some drafts concerning the black box tests, hence a list of some important inputs to give to the particular case of study
- some cases of intermediate integration are rolled out. We can find the integrations from the edges of the system to the most internal parts.
- User experience is linked with the Mobile App and the Car System and then these entities are attached to the System. the same thing can be said for the Payment System and the Database
- System is whole integrated and tested. This last steps can take into account the top down strategy: when the integration produces some undesired results the decision can be starting from the "top", going through the System until the bug is not found

Testing
Internal API

- for the execution of the steps discussed in the previous slide the ITPD document also refers to the fact that a solid documentation of the input methods used for reaching the various functionalities is indispensable
- some APIs are shown in the document
- the use of these APIs are foundamental, not only in the phase of testing, but with the optic to have a reusable and expandable vision of whole the system implementation: a future developer can refer to the API for improving the case of tests, resolving bugs, in general maintaining and enlarging the code in case of necessity, after the release of the Application

Project Planning

Cost estimation

Approach

- We used the Function Point method together with COCOMO II to estimate the project’s effort and duration
- We computed every estimation twice, to get a lower bound and an upper bound for all estimates

Cost estimation

Results

Lower bound

- UFP = 50
- KSLOC = 5.2
- Effort = 18.32 person-months
- Duration = 9.3 months

Upper bound

- UFP = 70
- KSLOC = 7.28
- Effort = 26.54 person-months
- Duration = 10.4 months

Project plan

Milestones and schedule overview

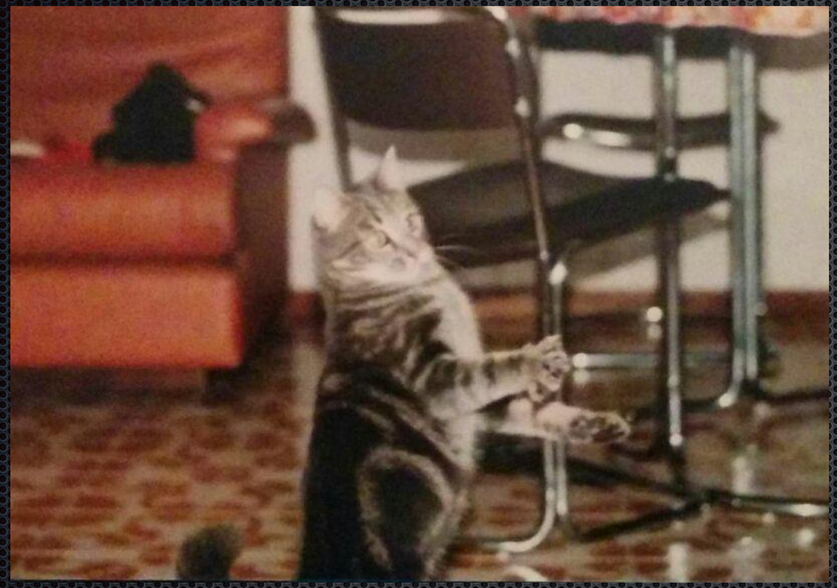
We developed a schedule spanning over 9 months

- RASD 13/11/16
- DD 11/12/16
- ITPD 15/01/17
- PP 22/01/17
- Development and testing 15/07/17
- Deployment 31/07/17
- Maintenance ...

Risk evaluation

- Business risks
- Implementation behind schedule
- Loss of application data
- User data leaks

Thank you for your attention



cat tax

Luca Marzi
Valeria Mazzola
Federico Nigro