# Design Document

November 23, 2016

**Document Version 0.3**

Matteo Frosi (mat. 875393)
Luca Costa (mat. 808109)

# Contents

# 1 Introduction

## 1.1 Purpose

This document, addressed mainly to developers, aims to go into further details regarding the specification of the project, described in the RASD. Here more technical features will be described, including:

- The high level architecture
- Some possible design patterns
- The description of the main components and their interaction (Runtime View)
- A brief description of the algorithms on which the software relies
- A more detailed overview of the user interfaces

Although these are the main topics of the document, other minor details will be touched and discussed, such as the architectural style and a brief comparison with other styles and the mapping between the requirements defined in the RASD and the designed elements described in this document.

## 1.2 Scope

PowerEnJoy is an electrical car sharing service, based on a mobile application. The targets of the service, intended as users, are people that needs to move from a place to another within a city and requires a conveyance to move (because they don't have their own or simply can't use it).
A user can make a reservation for a car, using the mobile app and his/her account, and check for the availability and status of all the cars within his/her position, identified using GPS localization, or a specific one, inserted manually by the user. As stated before, to access the service, the user must possess a private account, so a registration is needed.
The system provides the users a safe way (identification code) to access the cars, and the riding service and keeps trace of the status of all the cars.
Moreover, the system prizes or punishes a respectively good or bad behavior from the users, applying a discount or an overcharge on the cost of a ride. As example, if the user leaves the car without much battery, he/she will have to pay more than the standard cost of the ride, because the car will need to be charged and this operation has a cost. On the other hand, if a user plugs the car before ending the service, it receives a discount.
The system includes other functionality, such as GPS based maps available in every car, an emergency procedure in case an accident occur during a ride and the notification of a car status if the user requested it.

## 1.3 Definitions, acronyms, abbreviations

- RASD: document about the requirements analysis of the project.
- DD: document about the design choices and the components description of the project.
- GPS: global navigation satellite system that provides location and time information in all weather conditions, anywhere on or near the Earth where there is an unobstructed line of sight to four or more GPS satellites.

- SMS: short message service; it is a notification sent to a mobile phone, we need a GSM gateway to use it.
- GMS gateway: device that allows SMS text messages to be sent and/or received by email, from Web pages or from other software applications by acquiring a unique identifier from the mobile phone's Subscriber Identity Module, or SIM card.
- Push notification/ push message: it is a notification sent to a smartphone using the mobile application.
- API: application programming interface; it is a common way to communicate with another system.
- REST: representational state transfer, it's one way of providing interoperability between computer systems on the Internet.
- RESTFul: REST with no session.
- UX: user experience design
- URL: uniform resource locator.
- MVC: model view controller, it's a design pattern.
- SOA: service oriented architecture, it's a style of software design where services are provided to the other components by application components, through a communication protocol over a network.
- Layer: way of organizing code in sections sharing a common goal. The highest partition includes Presentation Layer, Business or Logic Layer and Data Layer.
- Tier: physical deployment of layers.
- Safe area: it is a specific area where the electric cars of PowerEnJoy service can park. The set of safe areas is pre-defined and owned by the company/society that requested the management system for the service.
- Special safe area: it is a safe area where power grid stations are installed.
- Power grid station: it is an installation that allows the recharge of an electric car.
- Communication primitives: set of instructions and procedure that allow a communication between machines and devices over a network.
- Sensor data retrieval: procedure that consists in getting all the information collected by the sensors of the car.

## 1.4 Document structure

- Introduction: this section introduces the design document. It gives an overview on what topics will be covered and what aspects described in the RASD will be improved here.
- Architecture Design: this section is divided into different parts, each describing an aspect of the software design.
  - Overview: brief description of the division in tiers and layers of the application.

  - High level components and interaction: high level view of the components of the application and the way they communicate.

- Algorithm Design: this section describes some of the algorithms that the application will rely on. To focus on the algorithm idea and not the fine grained implementation, pseudo-code will be used.
- User Interface Design: this section presents mockups and user experience explained via UX diagrams.
- Requirements Traceability: this section aims to explain how the decisions taken in the RASD are linked to design elements.

# 2 Architecture design

## 2.1 Overview

PowerEnJoy relies on a three tier architecture. Referring to the proposed system architecture presented in section 2.3 of the RASD, the following figure represents the tier division of the system.

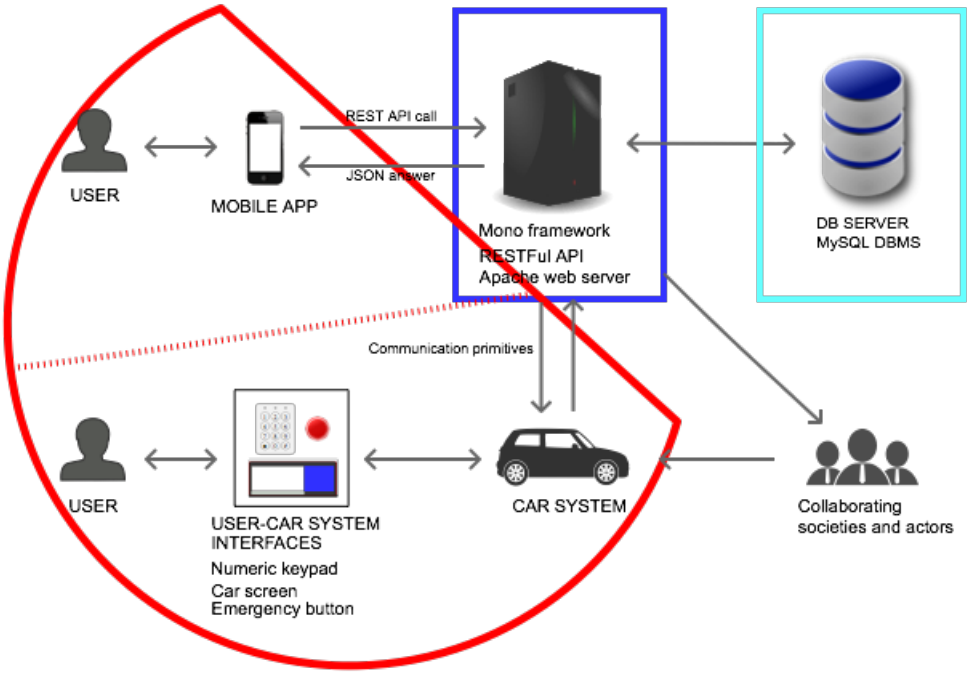


Figure 1: Proposed architecture, tier division - red: client, blue: server, azure: database

The client side includes all the possible ways by which a user can interact with the server (and the service), that are the mobile application and the car itself. The first represents the classical idea of client tier, including a GUI and a minimal amount of processing power that allows the user to have a connection with the rest of the tier, shown in a graphical way, that is more user friendly. The latter is, instead, a particular client tier that acts as a mini server, because it contains more logic than the mobile application. Here there is not a real graphic interface but multiple mechanical interfaces (MUI) can be detected, such as the numeric keypad that make the user able to unlock the car or end the service, the emergency button and the car screen, that is thought more towards an informative interface than an interactive interface. To better understand this concept, follow the same figure, this time divided in high level layers.

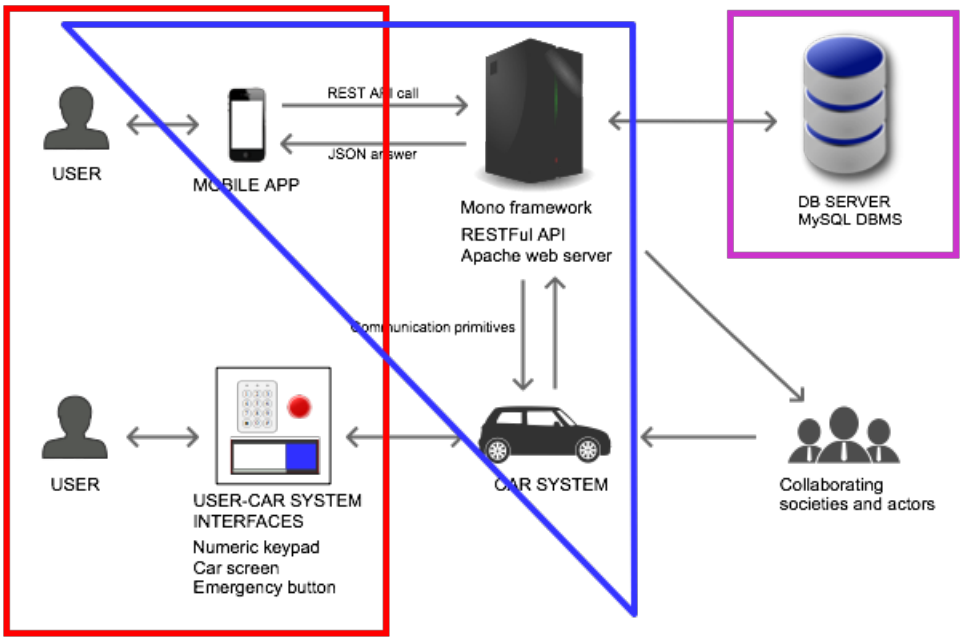


Figure 2: Proposed architecture, layer division - red: presentation, blue: logic/business, purple: data

The mobile application is also part of the logic layer because it can hold some immediate processes, instead of delegating them to the main server. An immediate example could be the elaboration of the GPS based map during the research of an available car.

## 2.2 Overview of the car system

It should be useful to spend some words to discuss more in detail the structure of the car and the way the user can interact with it, and consequentially with the server. Using multiple sensors, information about the car status can be retrieved, such as battery charge, number of passengers, degradation level of the car components, or even the localization data, obtained with GPS. Such data pool is interfaced with the car system, that resembles the shape of an application. The car app can be considered as a cluster of procedures and interfaces that allows user, car hardware and system to communicate and "know" about each other. Speaking in pattern terms, the car app emulates the controller entity in the MVC pattern, even if only partially. As said in the previous subsection, while the emergency button and the numeric keypad make the user interact with the car in an active way, the car screen acts as a passive element and is simply needed to inform the user of the car status and the service info. We can consider such feature as a way to diminish the interaction user-system to a very strict level, without changing the user perspective of the service. Such reduction lightens the logic needed in the car, making the project more achievable from a budget and complexity point of view.

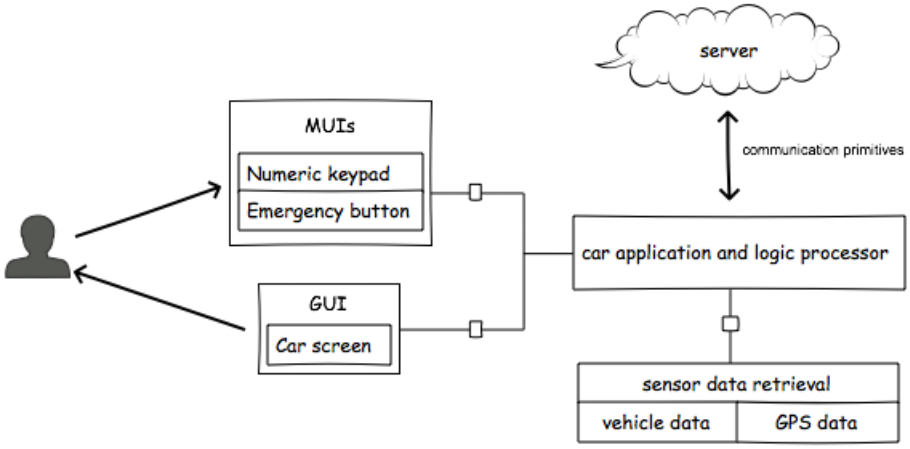


Figure 3: High level car system

# 3 Document further information

## 3.1 References

- RASD, before Version 4.5
- Assignments AA 2016-2017.pdf
- `https://msdn.microsoft.com/en-us/library/ee658116.aspx`, containing some information about layers design
- `http://alistair.cockburn.us/Hexagonal+architecture`, contains some information about the Hexagonal architecture
- `https://www.nginx.com/blog/introduction-to-microservices/`, contains some information about the Microservices architecture
- `https://8thlight.com/blog/uncle-bob/2011/09/30/Screaming-Architecture.html`, contains some advices over the architecture choice and implementation
- `http://www.uxbooth.com/articles/8-must-see-ux-diagrams/`, contains some examples of user experience diagrams
- Sample Design Deliverable Discussed on Nov. 2.pdf
- Paper on the green move project.pdf
- Second paper on the green move project.pdf

## 3.2 ChangeLog

- [21/11/2016] [**Version 0.1**] :: Added first part of the document introduction, including preface, Purpose, Scope and Definitions. ChangeLog and Hours of work sections also introduced.
- [21/11/2016] [**Version 0.2**] :: Added the References and Document structure sections.

- [23/11/2016] [**Version 0.3**] :: Added the overview of the architecture of the project and a brief description of the car system, so sections 2.1 and 2.2

## 3.3 Hours of work

**Matteo Frosi**

[Before 21/11/2016]: 3.00 hours (spent in analyzing well known architectures, to study and apply them at our problem).
[21/11/2016]: 2.00 hours (further analysis of an applicable architecture to our problem, alleging to the classic 3 Tier architectures).
[21/11/2016]: 1.00 hours (writing of the first part of the document).
[23/11/2016]: 2.00 hours (writing of the overview section of the document and drawing of architectures images)