Politecnico di Milano

PowerEnjoy Service - Design Document

December 4, 2016

Version 1.1

Authors:

- Domenico FAVARO (Mat. 837995)
- Matheus FIM (Mat. 876069)
- Caio ZULIANI (Mat. 877266)

Prof. Elisabetta DI NITTO

Contents

1 Introduction

1.1 Purpose

This Design Document serves the purpose to present to all parties interested the description of the structure for the PowerEnjoy Service. It provides documentation of Software, Architecture and other important aspects of Design to help the understanding and development of the System. Every component that is implemented for the System will be explained as well as the purpose they serve to contribute to the fullfilment of all the project Requirements. All strategies and design decisions will be documented as well. Software Design Description, decisions and key information to be used to communicate the general purpose of the structure to our stakeholders, and serve as detailed documentation for the components of the System for the developers.

1.2 Scope

This Document presents the PowerEnjoy Service System, an electric car sharing service. To better understand the broader scope of the service, it is presented in the RASD Document. This Document will not include detailed information on all the tools and protocols that will be used in the development of the System but rather their purpose and functionality inside the PowerEnjoy System. As example, general knowledge of the Client-Server structure is expected as it will not be rigorously explained but instead how such structure will be used to satisfy our System's Requirements.

1.3 Glossary: Definitions, Acronyms, Abbreviations

- **RASD:** Reqirements And Specifications Document.
- **DD:** Design Document.
- **Java EE:** Java Enterprise Edition. Software Development Platform.
- **App:** Application. Refers to the deployed service as Web or Mobile Application.
- **Component:** Software element that implements and offer functionalities in the System.
- **EJB:** Enterprise Java Beans. Component in the Business Tier for the Application Logic.

- **DBMS:** DataBase Management System.
- **JDBC:** Java Database Connectivity, Java API to connect to DataBases.
- **HTTPS:** Hypertext Transfer Protocol over TLS. Protocol to safefully communicate over the Internet.
- **TLS:** Transport Layer Security that provides communication security.
- **REST/RESTful:** Representational state transfer, architectural style for the System communications.
- **MVC:** Model View Controller, software design pattern for implementing interfaces.
- **API:** Application Programming Interface.

For other concepts concerning the Service definition look in the **Glossary** section of the RASD.

1.4 Reference Documents

- Specification Document: Assignments AA 2016-2017.pdf
- PowerEnjoy Requirements And Specifications Document (RASD)
- IEEE Std 1016-2009 IEEE Standard for Information Technology-Systems Designs-Software Design Descriptions (SDD IEEE 1016-2009.pdf)
- Example Documents:
  - Sample Design Deliverable Discussed on Nov. 2.pdf
  - Software Design Document (SDD) Template (sdd_template.pdf)

1.5 Document Structure

Section 1 - Introduction: This section provides a general description of the purpose and structure of this Design Document.

Section 2 - Architectural Design: This section illustrates a broader to specific view of the components that form part of the System, presenting from the overview of the architecture of the system to a description of how each component will interface inside the structure.

Section 3 - Algorithm Design: Important functionalities of the System that require the development of algorithms will be described in this section.

Section 4 - User Interface Design: All details refering to how the User will interface with the System, from the Web Application to the screens inside the Cars will be shown in this section. As well as general mockups of the Graphical User Interface (GUI) screens.

Section 5 - Requirements Traceability: In this section is explained how the Design decisions and structure help fulfill the Requirements for the System that were defined in the RASD.

Section 6 - Effort Spent: Detailed record of the hours worked for each member of the team is documented in this section.

Section 7 - References: Any reference to additional external sources that can help the better understanding of this Document is documented in this section.

2 Architectural Design

2.1 Overview

As was show in the Proposed System part of the RASD, the PowerEnjoy structure will consist on a 3-tier Client-Server Architecture. The server part built on a JEE Platform with access to the Company's Database Server and the Client consisting of Users, CRM and Cars that will use a browser to access our Web Application and a Mobile Device and Screen inside the Cars will access the Mobile Application. As we need to interface our application with the Car functions (i.e. Lock, Unlock, Battery state) the application must have some Logic implemented client side. This distributed Logic structure will allow GUI to be created on the Client side as well. The Database Server will belong to the Company and will allow registration of User, Reservations, Rides and Payments. Cars, CRM Employees and Parking Locations are asumed to be added outside of this System.

- Client Tier (Web App, Mobile App, Car)
- Server Tier (Java EE Web and Application Server)
- Enterprise Information System (EIS) Tier (DataBase Server)

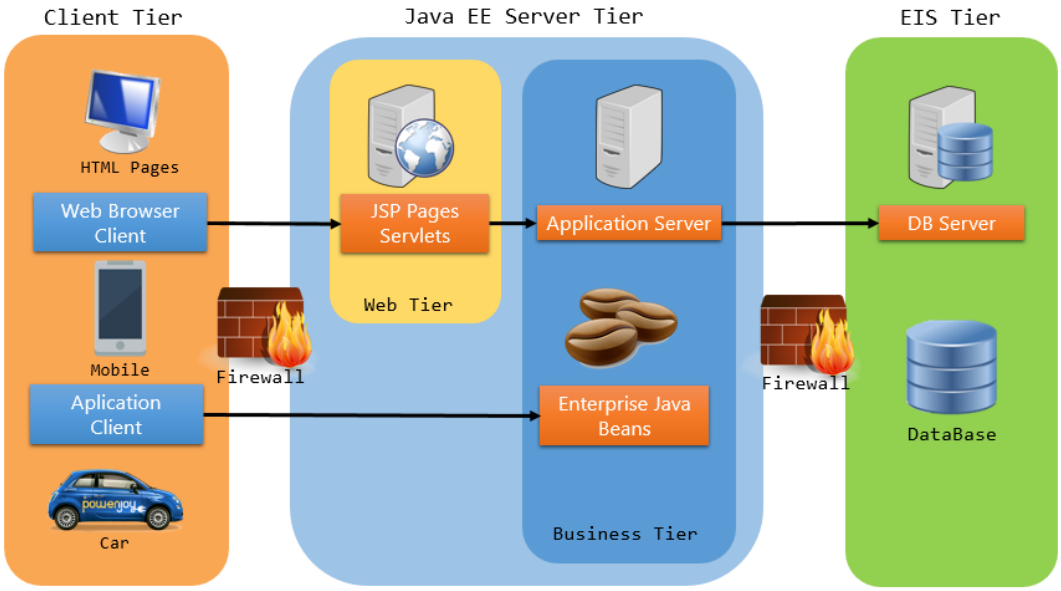


Figure 1: Proposed System Architecture

2.2 High level components and their interaction

Following our Architecture we organize the High level components on the 3 Tiers. This overview shows the interaction between the components and the entities they represent inside the System. Clients will have their User and CRM Components, distributed as Interfaces and Web Pages for the Apps. Inside the Server through Servlets and Session Beans a Session Manager will provide the services to the clients by communicating with the respective Controllers. Based in the MVC patter Controllers will manage the data that will be viewed by the Client. Each Controller will manage and represent the System Entities previously explained in the RASD and will interface with a Persistent DataBase Connection to the DBMS. Helpers and Extra Feature Components are represented as well.

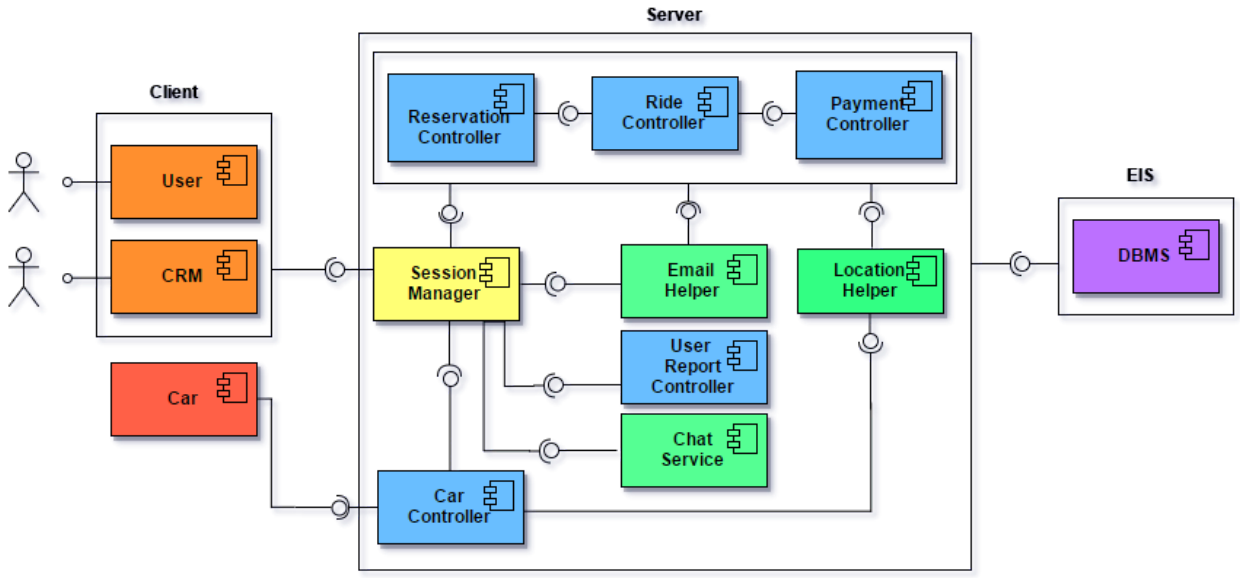


Figure 2: High Level Component Structure

2.3 Component View

2.3.1 User Component

Represents the User in the Client Tier that translates user actions and send requests to the Session Manager Component, then presents the output to the User. Is composed of several pages that present different types of information. Implement the Service Requests for each Actor, Web Pages in Web Server and Mobile Application and will have to interface with the central Server Component.

2.3.2 CRM Component

The CRM in the Client Tier will access just via browser, web server to the System. Is composed of a different set of pages and the Session Manager provides different services to it.

Figure 3: Client Components

2.3.3 Car Component

Represents the Car outside of our system that presents an interface to provide information and receive commands from the Car Controller Component.

2.3.4 Reservation Controller

Manages the Reservations made by Users, when a reservation is corfirmed, it will create the correspondant Ride.

2.3.5 Ride Controller

Manages the Active Rides. When a ride is finished, it creates a Payment using the Payment Controller.

2.3.6 Payment Controller

Implements the Logic to make a Payment or Transaction to a User Account, used by the Ride and the CRM Session Controllers.

2.3.7 User Report Controller

Used by the CRM Session to generate User Reports.

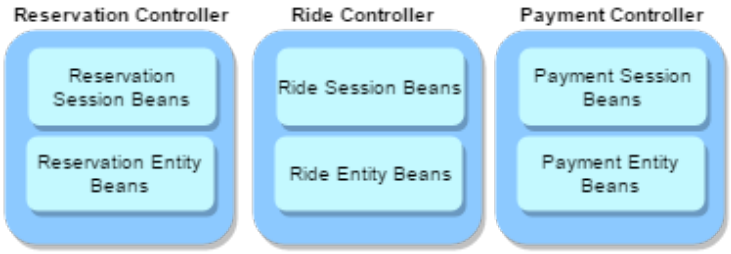


Figure 4: Entity Controllers

2.3.8 Session Manager

The Session Manager is a Macro component. Will expose an API for User and CRM Clients. Will handle the main functionalities for them and will redirect requests to their respectve Controllers. Inside, the CRM and User Controllers will handle the CRM and User sessions respectively.

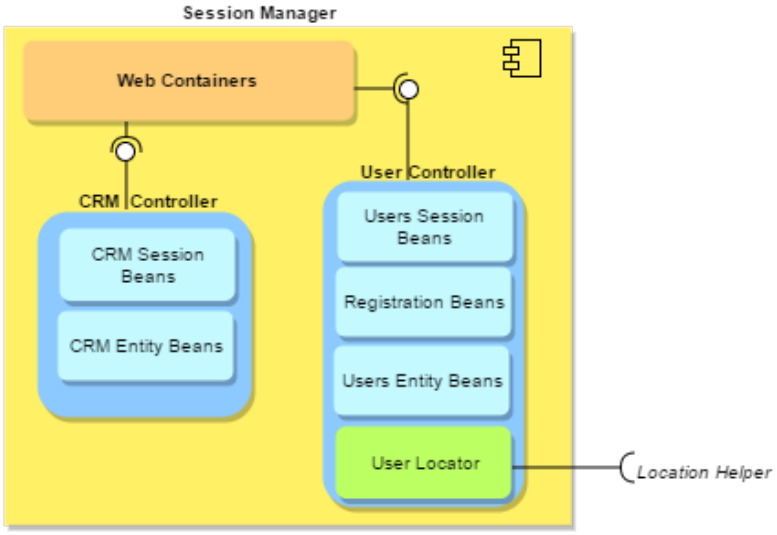


Figure 5: Session Manager

2.3.9 Car Controller

Will manage Cars Status, Locations and interface to Cars to send them instructions.

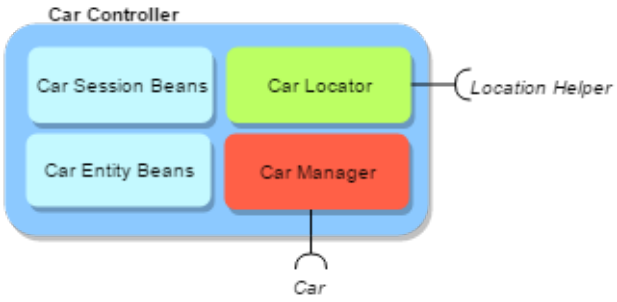


Figure 6: Car Controller

2.3.10 Location Helper

Will provide an interface for Location queries. Can interface to an external Service Provider API (Google Maps).

2.3.11 Chat Service

Will provide the chat platform to provide a channel of communication between Users and available CRMs

2.3.12 Email Helper

Will allow the System to communicate via mail with the Users (password, payment).

2.3.13 DBMS

Each Controller will implement a Persistent Entity for each class that will be mapped on the DB. The Structure of the DB will mirror the Classes in our System.

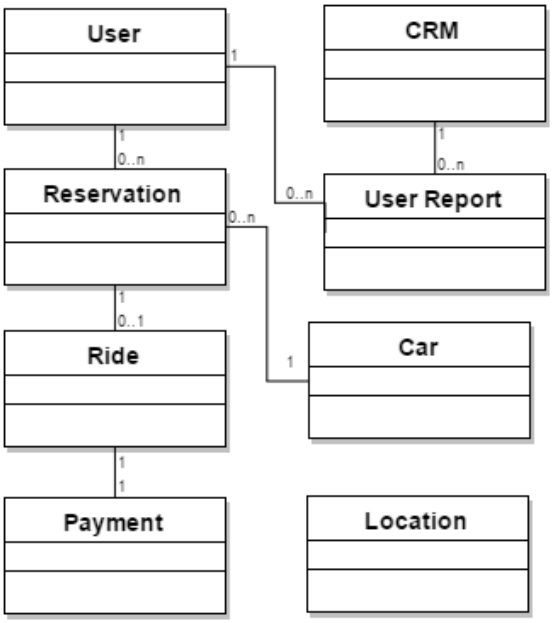


Figure 7: Database Structure

2.4 Deployment View

A first proposal for the Deployment View shows our main components in their deployed hardware as seen in the proposed Architechture. Mainly our Web, Application and DB Servers and the Client Devices. Our proposal for the communication protocols include:

- The Web Clients communicate to the Web Server via HTTPS.
- The Application Server exposes a RESTful API for the Web Server, Mobile Apps and Cars.
- The Application Server communicates with the DB Server using JDBC connection.

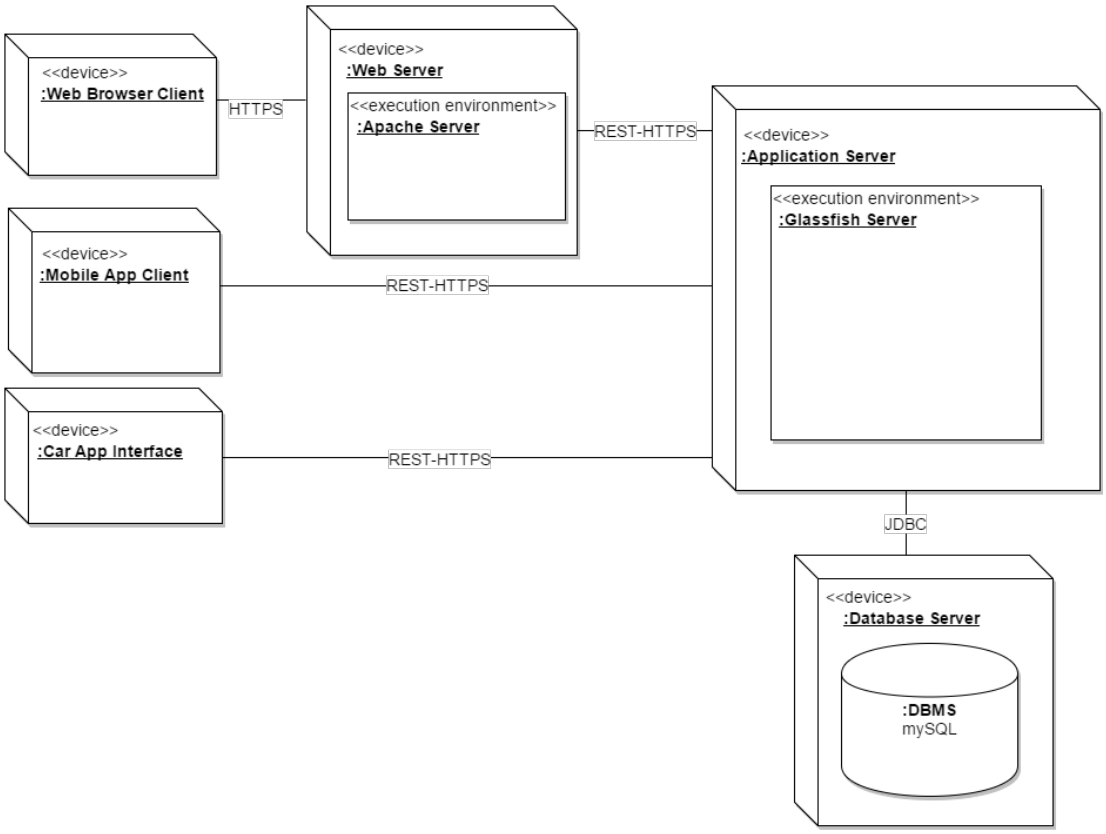
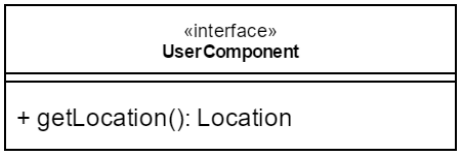


Figure 8: Deployment View

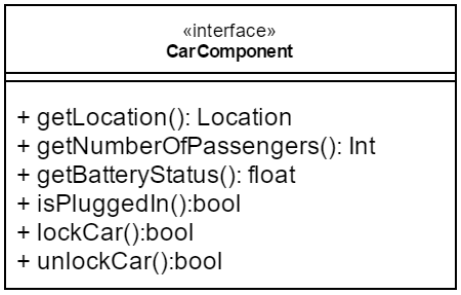
2.5 Runtime View

2.6 Component Interfaces

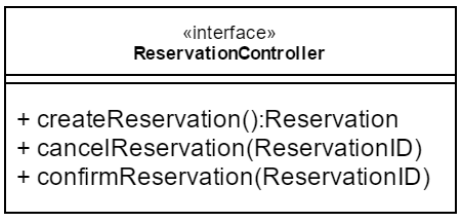
2.6.1 User Component



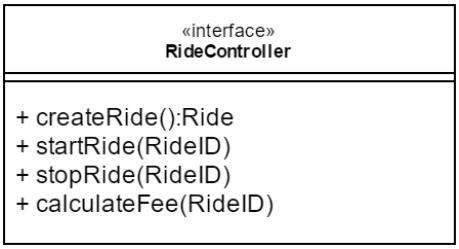
2.6.2 Car Component



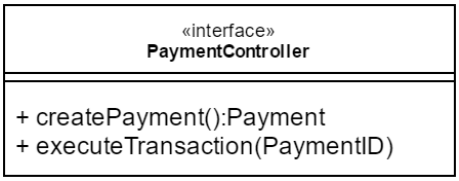
2.6.3 Reservation Controller



2.6.4 Ride Controller



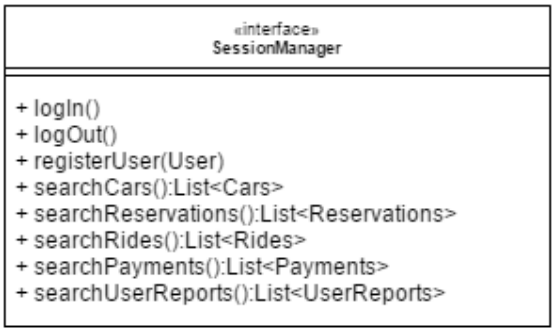
2.6.5 Payment Controller



2.6.6 User Report Controller

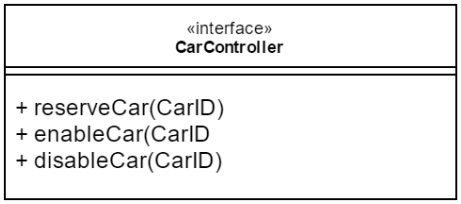
2.6.7 Session Manager

Will Interface with the Clients and answer requests redirecting them to the corresponding Controller.



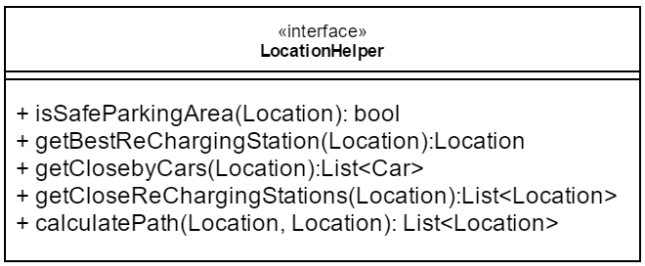
2.6.8 Car Controller

The Car Controller will interface directly to the Car so as well to communicate the Car methods will offer these other methods.

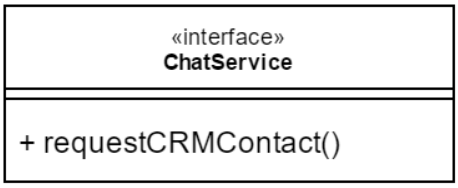


2.6.9 Location Helper

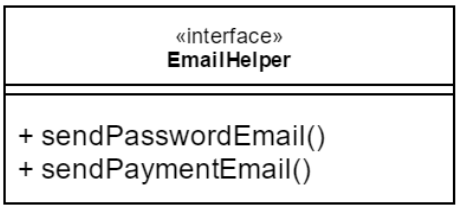
Location Helper will Implement our Location search Algorithms. It will help the System get the best ReCharging Station to ensure the best distribution of Cars in the city, as well as calculate Path and seach for nearby Cars or ReCharging Stations.



2.6.10 Chat Service



2.6.11 Email Helper



2.7 Selected Architectural Styles and Patterns

2.7.1 MVC 3 Tier Architecture

The 3 tier Architecture will be based in the MVC pattern as the Client represents the View, the Controllers will be implemented in the Application logic and the Model will be present in the Data Layer. This design allow us to present the data to our Users in a consistent and reliable way.

2.7.2 Client-Server Architecture

As the abstraction of our System is providing a service, the logical choice is to present the Client-Server model. The logic of our System implemented in our Servers and the Clients (Users and CRMs) will communicate with them to access our service. The simplicity and effectiveness of this model facilitates the deployment. Contrary to peer-to-peer

2.7.3 Communication: REST, HTTPS, JDBC

- Basing our communication patterns on REST will allow simplicity for the Client requests, mantaining persistence on our data, this helps keeping the communication stateless.
- REST over HTTPS is the most common practice and HTTPS is as well the standard secure protocol of communication on the Internet.
- Since we are deploying our App with Java EE, JDBC is the Java standard of communication with the DBMS.

2.8 Other Design Decisions

3 Algorithm Design

4 User Interface Design

For our User Interface (UI) we'll offer a mobile App for Users and a desktop web App for Users and CRM. They will offer:

- **LogIn Page:** First screen of the app with the LogIn and SignIn options.

Figure 9: UI LogIn Page

- **Main Page:** Main page of the app where the user sees the map his location and selects the available cars. In Case of CRM, it can see all Cars.

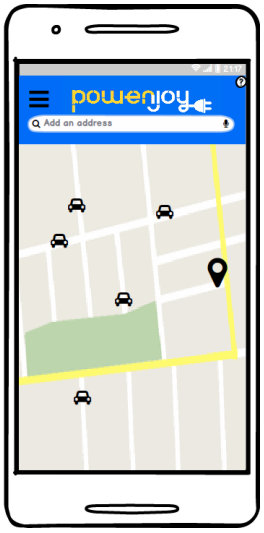


Figure 10: UI Main Page

- **Car Details Page:** Once a Car is selected this page allows to reserve a car and contais informations for the decision.

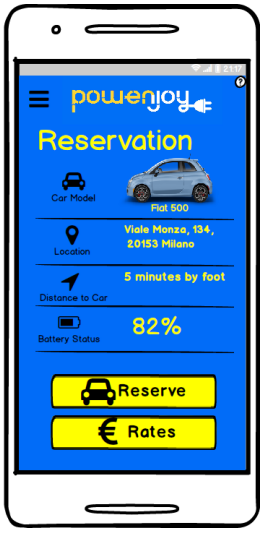


Figure 11: UI Car Details Page

- **Reservation Page:** Once the Reservation is made, the User can see the details of her/his Reservation and have the option to cancel it.

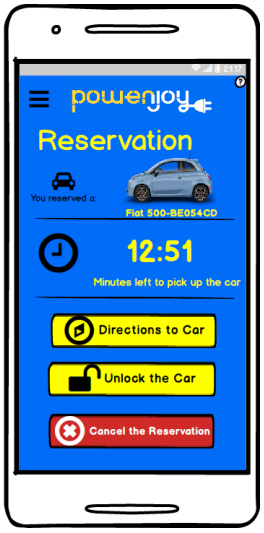


Figure 12: UI Reservation Page

- **Ride Page:** Once a Ride started this page shows the details of the current ride, it's present also in the Car screen App.

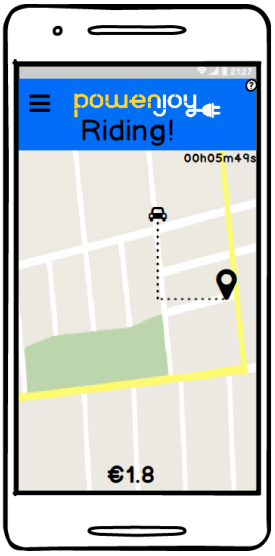


Figure 13: UI Mobile Ride Page

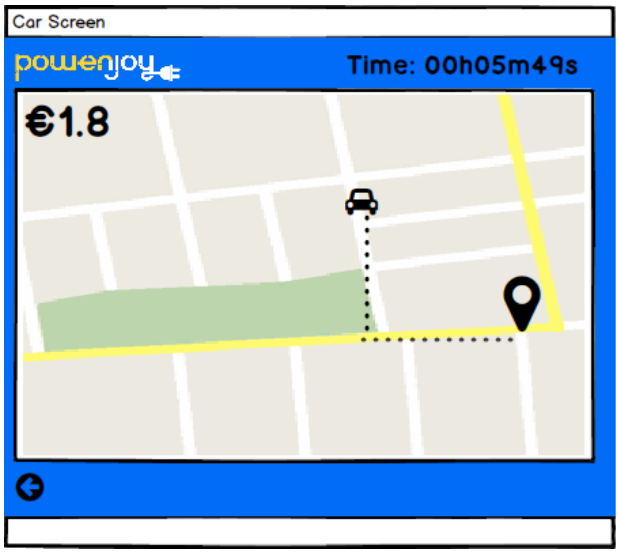


Figure 14: UI Car Ride Page.

- **Email Page:** While not part of the App UI, the System should respond with an email to the User at the end of the Ride showing the final details for it.

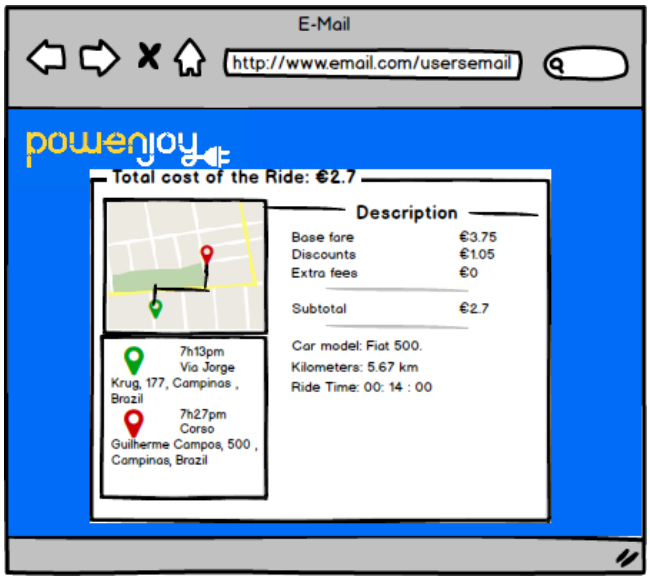


Figure 15: UI Email Page

- **Contact Page:** Page where the User can contact the CRM operator.

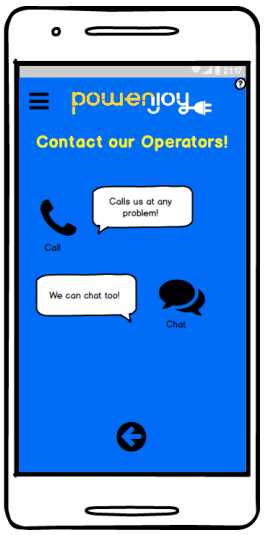


Figure 16: UI Contact Page

5 Requirements Traceability

6 Effort Spent

Date	Domenico	Caio	Matheus
27/11/16	1h	1h	1h
28/11/16	2h	-	-
29/11/16	2h	-	-
30/11/16	3h	-	-
31/11/16	1h	-	-
01/12/16	-	1h	-
02/12/16	5h	-	-
03/12/16	2h	-	-
04/12/16	2h	-	-
05/12/16	-	-	-
06/12/16	-	-	-
07/12/16	-	-	-
08/12/16	-	-	-
09/12/16	-	-	-
10/12/16	-	-	-
11/12/16	-	-	-

7 References

8 Used Tools

Keeping track of the Tools used during develop the DD document were:

- **GitHub:** for Version Control
- **Dia Diagram Editor:** for UML Diagrams
- **TeXworks:** for LaTex editing of this Document

9 Changelog

As the project and design decisions may change during the development this document is also prone to change. We'll document every version in this part.

- **Version 1.1:** 11/12/2016