Software Engineering 2 Project
Design Document

# PowerEnJoy


POLITECNICO
DI MILANO

Team: Nico Montali, Enrico Fini

# Contents

# Introduction

## 1.1 Purpose

The purpose of this document is to give a more detailed and technical overview about PowerEnJoy system.
This document is addressed to developers and aims to identify:

- The high level architecture;
- The design patterns;
- The main components and their interfaces provided one for another;
- The Runtime behavior.

## 1.2 Scope

This document will present different level views in order to describe clearly the architecture of PowerEnJoy System. In particular we will present the component view, both high and low level, the deployment view, the runtime view and a further description of user interface, analysed in its runtime flow.
The System is based on two mobile applications and a web application.
As already explained in the RASD, the targets for the Service are:

- The Drivers;
- The Workers;
- The Admins.

The System allows the Drivers to reserve cars from both the mobile application and the web portal. At the same time it manages the the status of the cars and, if necessary, assigns tasks to the Workers via an ad-hoc mobile application. The Admins of the System can interact with the web portal to make privileged actions.

## 1.3 Definitions, Acronyms and Abbreviations

**Driver** Client of the system, i.e. the one that uses the service, reserves and drives. Every Driver must provide personal informations (name, surname, email, birthdate, a valid license and payment information;

**Ride** A single usage of the service, starts when the user turn on the engine, stops when the car is locked inside a Safe Area;

**Blocked driver** A driver that has an invalid license or invalid payment coordinates;

**Worker** Employee of the company, perform physical actions (moving, charging etc) on cars;

**Report** Car issue reported by the Driver during a Drive. Will later be assigned to a Worker;

**Task** Piece of work assigned to a worker. Different type of tasks are described later. Every task assigns a single car to a single worker;

**Administrator** Employee of PowerEni that is in charge of the administration of the system through the administration console;

**Drop off** The act of leaving the car inside a safe area. This ends the service provided to the driver and effectively make the driver pay;

**Safe Area** An area in which the Driver can park the car and stop the Service;

**RASD** Requirements Analysis and Specifications Document;

**DD** Design Document;

**API** Application Programming Interface;

**DBMS** DataBase Management System;

**UI** User Interface;

**REST** REpresantional State Transfer.

## 1.4 Reference Documents

- The RASD we released before;
- Sample Design Deliverable Discussed on Nov. 2.pdf.

## 1.5 Document Structure

**Introduction:** contains an overall description of the content and the structure of the document;

**Architecture:** this section contains:

- **Overview:** coarse view of the System i.e. the division in tiers;
- **High Level Components:** description of the main components of the application and their interaction;
- **Component View:** more detailed view of the components of the application;
- **Deploying View:** this section shows the execution architecture of the system representing the deployed software and hardware.
- **Runtime View:** shows the behaviour and the workflow of the System during some typical situation;
- **Component Interfaces:** describes the interfaces between the components.

**Algorithm Desing:** we will present some algorithms with pseudocode to show the locig of the most important parts of the System;

**User Interface Design:** starting from the ideas we gathered in the RASD we will give a deeper description of the UI;

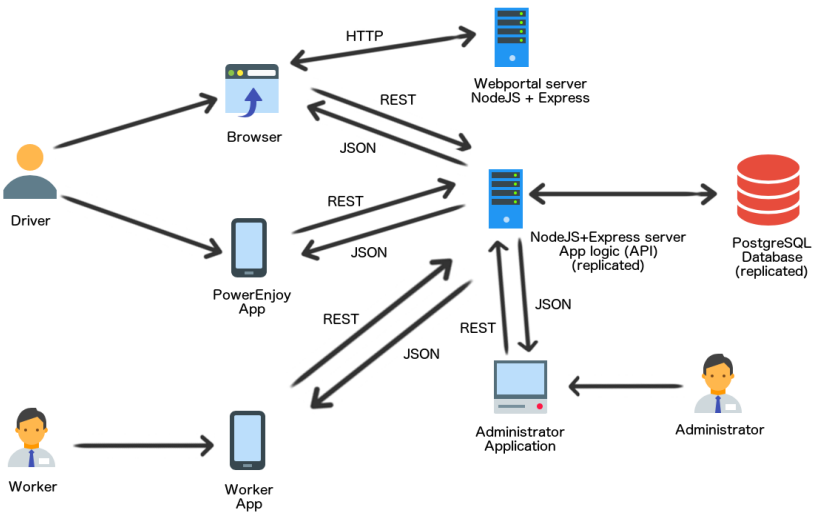
**Requirements Traceability:** An explanation of how our architectural choices respect the requirements described in the RASD.

# Architecture

## 2.1 Overview

The PowerEnjoy system is designed as a 3-tier application:

- **Data tier:** A PostgreSQL DB, deployed on AWS and replicated for availability and integrity.
- **Application tier:** A NodeJS application that provides APIs, deployed on AWS and replicated. The Express library is used to realise API endpoints. These APIs are exposed in a REST manner, responses are JSONs. The replication of the application grants fault tolerance and, using a load balancer, also availability.
- **Presentation tier:** The presentation layer is realised with the 2 mobile applications (Driver and Worker), the web portal (Driver) and the Administrator application. The web portal is realised using EJS templating (server-side) and JQuery (client-side).



## 2.2 High level components

The high level structure of the System can be divided into X components. The main component is the **core**, that receives requests from **clients**, either **drivers**, **workers** and **administrators**; depending on the client type, the component provides different interfaces. The core also communicates with the **DBMS** component, responsible of data persistence. The core implements all the business logic:

- Process drivers requests and allocates the requested vehicle
- Manages vehicles, track their state
- Manages users (drivers signup, login)
- Manages reports and automatic tasks then it allocates them to workers