

Version: 1.0.0

Release date: 25-11-2016

Riccardo Cattaneo 873647

Fabio Chiusano 874294

PowerEnJoy, Design Document

Table of Contents

1 Introduction 3

1.1 Purpose 3

1.2 Scope 3

1.3 Definitions, Acronyms, Abbreviations 4

1.4 Reference Documents 5

1.5 Document Structure 5

2 Architectural Design 6

2.1 Overview: High level components and their interaction 6

2.2 Component view 6

2.3 Deployment view 6

2.4 Runtime view 6

2.5 Component interfaces 6

2.6 Selected architectural styles and patterns 6

2.7 Other design decisions 6

3 Algorithm Design 7

4 User Interface Design 8

5 Requirements Traceability 9

6 Effort spent 10

7 References 11

# Introduction

## Purpose

This document describes the hardware and software architecture of the PowerEnjoy System. Therefore, it outlines hardware tiers and all parts of the software and how they will work and cooperate together.

In particular, this document contains information about:

* Architecture Design with related pattern used;
* Main components and software interface design;
* Runtime behaviour of the system
* User Interface Design

## Scope

PowerEnjoy is a car-sharing service based on a mobile web application.

It allows the user to see, thanks to the help of an external search-on-a-map handler, where the electric cars are, only if they are close to an address provided by either the user or his/her GPS Location.

Hence, it allows users to reserve an electric car and to get on board when he/she is close to it.

The car software takes into account the minutes of usage of the car, the number of passengers, the battery level and the location of release.

The system then calculates the charges the user for the ride.

The main purpose of the system is to create a new and smart car-sharing service, that incentivize virtuous and green behaviours.

## Definitions, Acronyms, Abbreviations

* DD: Design document.
* RASD: Requirements Analysis and Specification Document.
* JSE: Java Serial Edition.
* JEE: Java Enterprise Edition.

## Reference Documents

## Document Structure

This document specifies the architecture of PowerEnJoy spreading from the general into the specific. Also it describes the architectural decisions and tradeoffs and justifies them. The design was guided by a top-down process approach and the document structure reflects this tactic.

The document is organized as follows:

1. Introduction, provides a synopsis of the architectural descriptions.
2. Architectural design, provides a general description of PowerEnJoy including its  functionality and matters related to the overall system and its design.
3. Algorithmic design.
4. User Interface design.
5. Requirements traceability.
6. Effort spent.

# Architectural Design

## Overview: High level components and their interaction

### High level overview of components

We propose to make a web app that will give users a comfortable way to use our service. The reasons according to which we have chosen a web app instead of the web site are explained in the “Proposed system” chapter of the RASD document.

We propose the following components:

Presentation tier

Data tier

**Database**

**Server**

Presentation tier

**Mobile**

**Car**

Logic tier

The web app will be available to all the major mobile operating systems and will be developed in a way to communicate with the PowerEnjoy servers through API and Http requests, consequently the front-end will lie on the server. This decision brings to a trade-off between the app performance and the overall flexibility of the system in terms of UI and functional requirements, since, in the case with the front-end inside the app, we would have to make the user update it every time PowerEnjoy proposes new functionalities or a new UI.

On the server side, the Main Server (also simply called “Server”) contains two software tiers, the Web Tier and the Business Logic Tier.

Instead, the Database Server contains the data source, it is the database allowed to store all the relevant data and to retrieve them.

Therefore the Main Server interacts User Tier (the web-app) in order to provide PowerEnjoy Service to users, and with the Database in order to store persistent data.

Of course, the electric cars must be able to communicate with the server, so they must be provided with an Internet connection and an on-board computer that must be able to run Java software. However, the car is only an agent in our system and therefore all the business logic will lie on the server.

As for the communications, the server will expose a RESTful API to the mobile app and the electric car.



Database Server

Main Server

Car on-board

computer

API response

API request

eg: drive concluded

API request

eg: payment request

API response

API request

eg: reservation created

or

Http request

Mobile Web-App

**Payment hander**

**&**

**search-on-map**

**handler**

API response

or

Http response

### Technologies used

Here are the technologies we’ll use to implement our architecture:

* Phonegap: cross-platform development framework for mobile applications;
* JEE: server logic tier;
* MySQL: server data tier;
* JSE: car server;

### System operation examples

FROM TheraWii EXAMPLE:

- Description of the problem

- Technologies used

- System architecture (list components, informal)

- System operations (sequence diagram example, possibly involving all the components)

## Component view

High level class diagram of the main classes (that are the components, probably).

Diagram that shows the interactions between components.

## Deployment view

Deployment units:

- App

- Server

- Car program

Diagram that shows their interactions.

## Runtime view

Looooots of sequence diagrams.

## Component interfaces

Components class diagrams, very detailed. Text explanations of all classes: what the class does, what its methods do, design constraints, permormance issues…

## Selected architectural styles and patterns

Three tiers, event-based, MVC…

## Other design decisions

Maybe data model (class diagrams, Object Relationship diagrams) and data storage (ER diagrams, SQL stamements for tables creation).

# Algorithm Design

- GPS usage (not sure)

- Search for near cars:

1. App asks server for near cars and sends it its location;
2. Server retrieves cars locations quickly thanks to some spatial data structures (https://en.wikipedia.org/wiki/Spatial\_database);
3. Server sends the locations to the app;
4. App is happy.

- Search for near parkings: similar to search for near cars.

# User Interface Design

All the screens!

How to go from a screen to another.

Description of each screen.

Some scenario examples.

# Requirements Traceability

Table with the following columns:

- Requirement

- Description

- Design reference (all the references in this document that together satisfy the requirement).

# Effort spent

# References