POLITECNICO
MILANO 1863

11 December 2016

PowerEnJoy

Design Document

Blanco Federica 875487
Casasopra Fabiola 864412

Software Engineering 2 Project
2016/2017

Contents

List of Figures

List of Tables

1 Introduction

The main purpose of this document and the project's scope are described in this section. Moreover, we are going to give some definitions which will help the reader to better understand the content of this document and we are going to show the reference documents that have been used to redact this one. At the end, it will be explained the structure of this document.

1.1 Purpose

This document is called *Design Document* and, from now on, we will refer to it using the acronym DD. Its purpose is to provide a complete description of the system specified in the RASD, giving enough technical details to allow the proceeding of the software development. So, we must have a good understanding of which are the components of the system, how they interact, which is their high level architecture and how they will be deployed, highlighting the design patterns we decided to use. In addition, the RASD is useful for the developers in order to figure out how to implement the entire system, thanks to the general description of the architecture and the design of the system to be built. For this reason, it must be complete and correct as much as possible. The DD contains both narrative and graphical documentation of the software design, including, for example, user experience diagrams, entity-relation diagrams, component diagrams, and other supporting requirement information.

1.2 Scope

The aim of the project PowerEnJoy is to provide a car-sharing service that involves *only* electric cars. In this document, we will give more information about the design choices for the development of this application. In order to have more details about the scope of our project, you may refer to the *Section 1* of the Requirements Analysis and Specifications Document.

1.3 Definitions, acronyms, abbreviations

1.3.1 Definitions

1.3.2 Acronyms and abbreviations

Here there is the acronims and abbreviations list:

DD Design Document

RASD Requirements Analysis and Specifications Document

GUI Graphical User Interface

JEE Java Platform, Enterprise Edition

REST Representational State Transfer

EIS Enterprise Information System

JSF JavaServer Faces

UML Unified Modeling Language

API Application Programming Interface

DB DataBase

OS Operating System

1.4 Reference Documents

- Specification document: Assignments AA 2016-2017.pdf
- IEEE Std 1016tm-2009 Standard for Information Technology - System Design - Software Design Descriptions.
- Requirements Analysis and Specifications Document: RASD.pdf (`https://github.com/fabiola-casasopra/sw-eng-2-project/tree/master/RASD/RASD.pdf`)

1.5 Document Structure

Here is presented the stucture of the documtent, with a brief overview of each section. While the RASD is written for a more general audience, this document is intended for only people directly involved in the development of our application, as the software developers, the project consultants and the team managers. So, each person, depending on its role, can go directly and read to the section he finds more relevant.

Section 1 There is an introduction with this document's purpose and other general information about it.

Section 2 There is an overall view of our system, describing all the components from different points of view and highlighting their interaction. Moreover, there is an explanation about the selected architectural system and design pattern.

Section 3 Here there are presented the algorithm we think are more relevant for the development of the application. They are mainly described using a pseudocode implementation.

Section 4 There is a description of all the details about the structure of the Graphical User Interface. This section is useful for the reder to get an idea on how the final application will look like.

Section 5 There is an explaination of how the requirements defined in the RASD map into the design elements that have defined in this document.

Section 6 Here there are given additional information that may be useful to the reader, such as the tools used and the time spent to redact this document.

2 Architectural Design

In this section, we will show the proposed architecture for our system, that allow us to give a more complete and general idea of the entire system and a better view of the relation with external components.

2.1 Overview

Now we are going to present an overall description of the architecture of our system. We propose a 4-tier architecture, following the model of the Java Platform, Enterprise Edition architecture: the **client**, usually a web client or an application client, runs on the client machine. Instead the **business** code, which is logic that solves or meets the needs of a particular business domain (such as banking) runs on the server machine, as it happens for the **Web-tier**. The **Enterprise Information System-tier** includes enterprise infrastructure systems, such as the database and legacy systems, and it runs on a third dedicated machine.
For sake of simpicity, in this document the web client and the mobile application are treated as one entity. For this reason, each communication between server and client will pass through the Web-tier.
JSF technology is a server-side component framework for building Java technology-based web applications and we will use it for the dynamic web pages. As far as the communication with the mobile application is concerned, we will consider an implementation of the REST paradigm.

In the following part, we will give a more accurate and detailed decription of this 4-tier architecture.

- **Client-tier**: This tier component are the Application Clients and Web Browsers. They both typically have a GUI, since they interact with the actors.
- **Web-tier**: This tier component has the task to manage the requests sent by the Client-tier and to forward these requests to the Business-tier. In a similar way, the Web-tier elaborates the contents generated by the Business-tier and it sends these contents to the Client-tier in a way that this tier components can render the information received.
- **Business-tier**: This tier represents the core of the whole system. This tier components contein the logic that solves or meets the needs of a particular business domain such as banking, retail, or finance, is handled by Enterprise Java Beans. They receive data from client programs,

processes it, and sends it to the EIS-tier for storage. An Enterprise Java Bean also retrieves data from storage, processes it, and sends it back to the client program. All the application logic resides here under the form of Enterprise Java Beans and Java Entities. This tier is connected to the Database through a Java Persistence API.

- **EIS-tier**: This tier components are usually database and legacy systems, where the entire system stores the needed persistent information.

2.2 Component view

2.3 Deployment view

2.4 Runtime view

2.5 Component interfaces

2.6 Selected architectural styles and patterns

2.7 Other design decisions

3 Algorithm Design

4 User Interface Design

5 Requirements Traceability

6 Appendix

6.1 References

6.2 Effort Spent