Design Document - v1.0

Gianpaolo Branca Luca Butera Andrea Cini

POLITECNICO
MILANO 1863

Contents

Introduction

Purpose

The purpose of this document is to provide technical informations about the service we are going to deploy described in the RASD.

Scope

Our service is about electric car sharing to provide a valid option to public transportation for low length trips. It is provided via mobile application for the final users and via web application for the operators of the company. Users can download the app with the Android play store or with the Apple store. Operators can enter the system directly from a browser in the boundaries of the companies network after logging in.

Definitions, Acronyms, Abbreviations

Reference documents

- RASD
- Specification document

Architectural Design

Overview

We are going to build our system following these guidelines (appropriate reasons for each choice will be given in the next section):

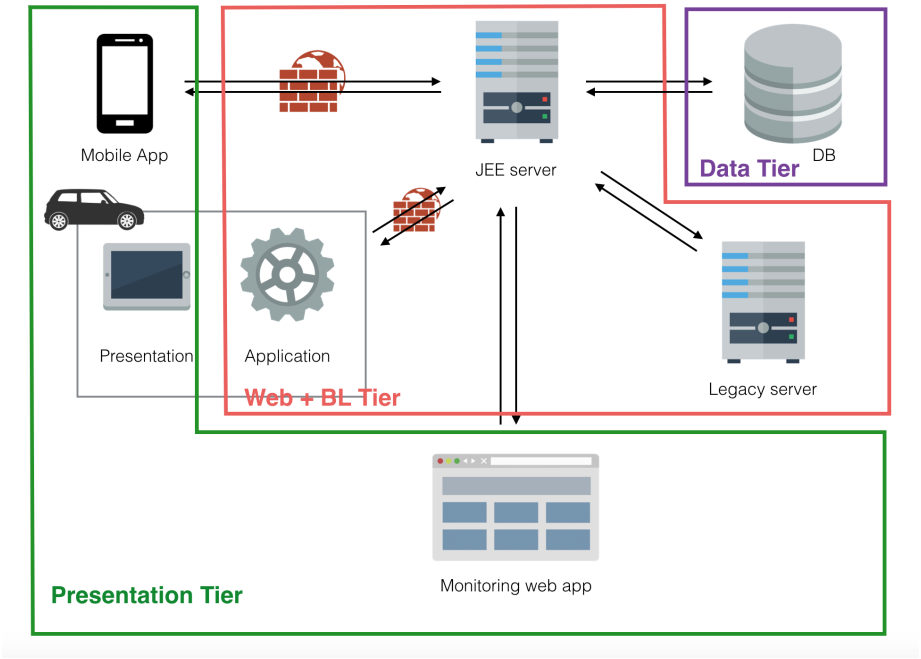
1. Our application will be implemented using a three-tier architecture as it is the most suitable(this point will be clear in the next steps) and maintainable one for our system.
2. For the mobile application the client side will be light-weighted, with only the presentation layer as there's no need to perform any kind of data manipulation on the user's mobile phone.
3. The car will be equipped with a general purpose system (with a stable Linux distribution as OS) with an application running on it, it will also implement some logic.
4. The operators will access the system through a web application.

5. Integration with the legacy server will happen trough its APIs for the purpose of providing maintenance to the cars and clients care.

Architectural choices

Server-side (Web , Business Logic and Data tiers)

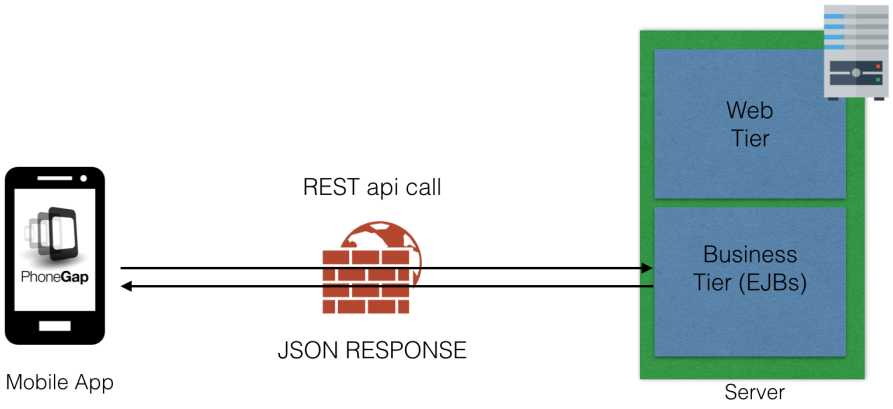
- We will develop our application and web server using the Java EnterpriseEdition framework (formally, using JEE, we should refer to our application as multi-tier, but for simplicity sake and because our system is distributed over client machines, JEE server machine and a database we will consider it three-tier) .
- JEE will allow us to shorten the development time and to achieve high performances while keeping our application complexity manageable.
- JEE makes our project use architectural structure that follows well-known best practices.
- We will use Oracle GlassFish Server (the commercial edition) as application server.
- GlassFish gives very good performance guarantees and is well supported.
- We will use the JAX-RS APIs to expose RESTful APIs with JSON that will be used client-side to interface with the web server.
- The usage of the RESTful standard will give our system robustness and flexibility.
- This will allow us to use Adobe PhoneGap to develop an hybrid multiplatform application for the client side.
- We will use the MySQL to manage our Database.
- We don't need advanced feature for data management that other DBMSs offer.
- MySQL is fast and easy to use.
- Free.
- Compatible with JDBC.



Client-side

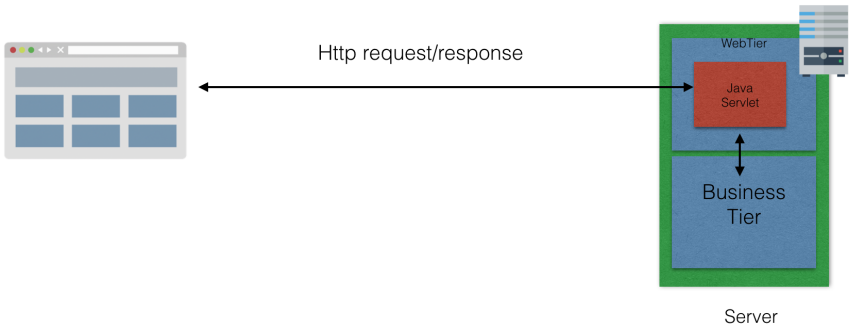
Mobile application

- We will develop our application using the standard web technologies (AngularJS, HTML, css) and use Adobe PhoneGap (built on Apache Cordova) to create a multi-platform mobile app. This approach will allow us to:
- Benefit from the experience of our engineers in web development.
- Reduce development time and cost.



Monitoring WebApp

- We will develop the monitoring Web using a JavaServlet.
- Easy to deploy.
- Integrated in the JEE framework.

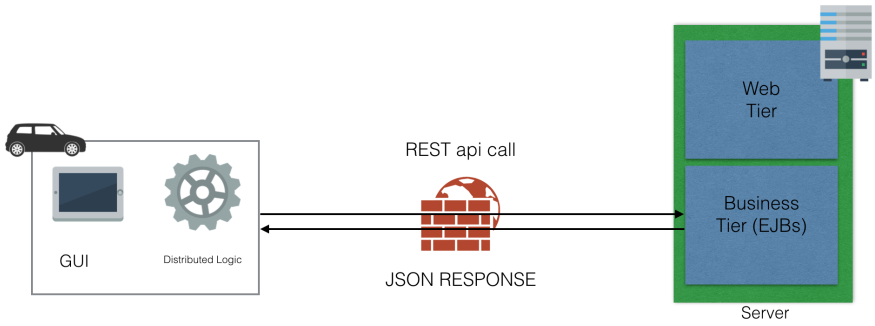


Car on-board application

- We will develop a Java application to run in the system of the car.
- We need a tool to have control over the car status.
- The application needs to contain not only presentation features, but also logic to elaborate the data coming from the sensors and manage the

execution of a ride without a continuos interaction with the server and deal with real time issues.

- The application will be able to retrieve informations from the car sensors (such as the battery level or the presence of mechanical problems)through OBD connector(Java libraries to read information from an OBD adapter already exist).



Component view

Deployment view

Runtime view

Component Interfaces

Selected architectural styles and pattern

These are the main design patterns that we are following in the design process:

- Model-Control-View : used pretty much everywhere, it's a really good choice of design that allow to keep very clear the role of every component of the system and that makes the system easy to deploy and maintain.
- Client-server : the staple good practice of a web based system.

Other design decisions

Algorithm design

User Interface design

Requirements Traceability

Effort spent