POLITECNICO

MILANO 1863

Contents

1 INTRODUCTION **1**
- 1.1 Purpose . . . 1
- 1.2 Scope . . . 1
- 1.3 Definitions, Acronyms and Abbreviations . . . 1
- 1.4 Reference Documents . . . 1
- 1.5 Document Structure . . . 1

2 ARCHITECTURAL DESIGN **2**
- 2.1 Overview . . . 2
- 2.2 High-level components and their interaction . . . 3
- 2.3 Component view . . . 4
- 2.4 Deployment view . . . 4
- 2.5 Runtime view . . . 4
- 2.6 Selected architectural styles and patterns . . . 4
  - 2.6.1 Utility tree . . . 5
- 2.7 Other design decisions . . . 5

3 ALGORITHM DESIGN **6**

4 USER INTERFACE DESIGN **7**

5 REQUIREMENTS TRACEABILITY **8**

6 EFFORT SPENT **9**

7 REFERENCES **10**

1 INTRODUCTION

1.1 Purpose

1.2 Scope

1.3 Definitions, Acronyms and Abbreviations

- **Software Architecture**: structure of a software system that is the result of a set of architectural choices.
- **Architectural Choice**: choice made to meet one or more design choices.
- **Design Choice**: choice made to meet one or more design issue. It is selected from a set of design option solving the design issue.
- **Design Issue**: any issue raised by functional requirements and non-functional requirements. Usually, a design issue has one or more design option that solve it.
- **Design Option**: a potential design choice that meet a specific design issue.
- **Layer**: logical level of separation used to spot and define the boundary of responsibilities of the content.
- **Tier**: physical level of separation used to spot the physical unit where the content will be deployed.

1.4 Reference Documents

1.5 Document Structure

2 ARCHITECTURAL DESIGN

2.1 Overview

In the following, the logical architecture of the system is briefly discussed, introducing the high level design choices made. With "logical architecture" here we mean how the system is logically structure, without implying any physical design choice (i.e: physical distinct allocation of web server and application server).

For a more specific and deeper discussion about the architectural styles and patters here introduced, together with reason for their choice, please refer to the Selected architectural styles and patterns section.

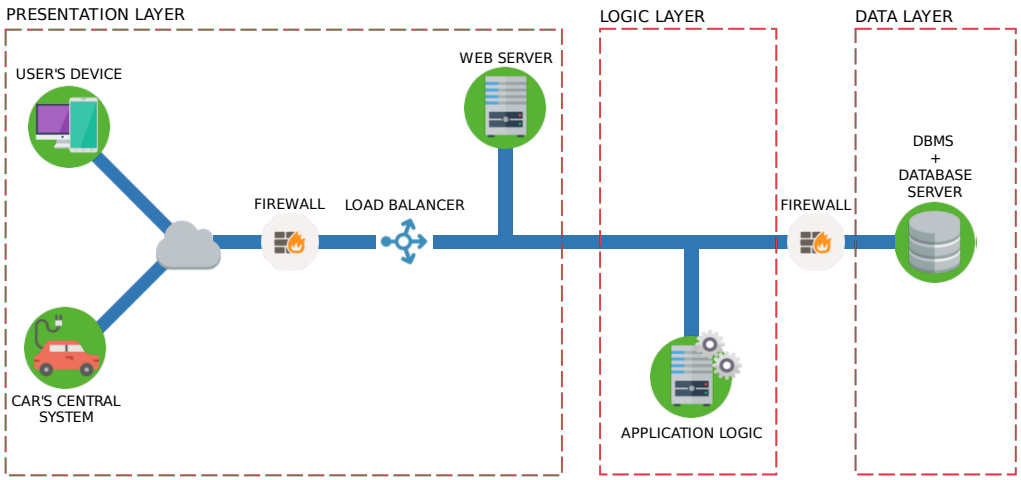

Figure 1: The system's high-level architecture

The general architecture of the system is based on the Client/Server paradigm. As depicted in Figure 1, a suddivision in 3 layers, also called Distributed Representation is chosen:

- **Presentation level**: the level where information requested are presentend (i.e: rendered by means of a web browser or simply delivered to the final user).
- **Application logic level**: here the whole logic of the system take place (i.e: vehicles' availability control).
- **Data level**: where the datas used, created and modified by the application logic tier are stored.

The elastic load balancer cloud computing patter is used and placed in front of the web servers and the application logic servers in order to increasethe availability of the system. To mitigate security issues, firewalls are located between Application logic tier and Datas tier and in between of the public internet and the system's private network.

2.2 High-level components and their interaction

The main software components forming the system, the interfaces provided and required from the software units, along with the relationship of necessity between components, are shown in the component diagram below [figure 2].

Note that not all the components represented, like the browser, are actually software units of the final system, but are part of the environment with which the system will interact.

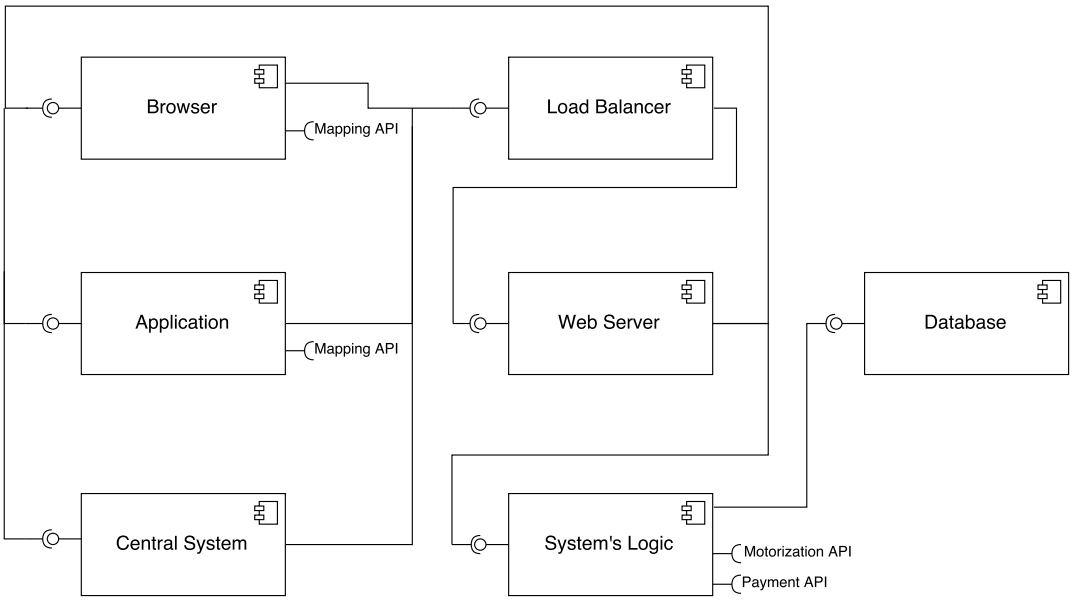

Figure 2: High-level component diagram

- **Browser**: The software used by the user to access to the service.
- **Mobile application**: the application installed on mobile devices and used by the user to access to the service.
- **Central system**: also called Car's Central System. It's the software responsible for car's management and interfacing with the system.

- **Load balancer**: the software used to distribute the workload to the servers.
- **Web server**: the software used to elaborate requests and send back responses.
- **System's logic**: the software responsible for the whole system's logic.
- **Database**: the software unit responsible for the management of queries and data.

2.3 Component view

2.4 Deployment view

2.5 Runtime view

2.6 Selected architectural styles and patterns

Here the main architectural styles and/or patterns are listed and described, motivating the reason why they have been chosen and how they integrate into the system.

- **Client/Server paradigm**: since most of the interaction between the system and the "outside" happens through requests and responses, the Client/Server paradigm result to be the most suitable design choice for the system. The paradigm is implemented through the introduction of web servers components that have the task of receive the incoming requests, process them and answer back with responses messages.
- **Distributed Representation**: in order to keep the client side as thin as possible, a client/server paradigm based on distributed representation is found to be the best solution possible for the project. All the business logic, the data and part of the presentation layer are placed on the server side, while the client side is loaded with part of the presentation layer.
- **4 Tiers**: the client/server architecture is split into 4 distinct tiers, decoupling data, business logic, request/response logic and client logic. In this way, the general maintenance of the system is dramatically improved, introducing an high modularity factor in the architecture.
- **Elastic Load Balancer/Elastic Component**: this cloud pattern is introduced in order to improve the general availability of the system. It is placed in front of the Web Server Tier and through the incoming requests and through the runtime information regarding the workload of the servers the Load Balancer allocate the workload in a balanced fashion.

2.6.1 Utility tree

2.7 Other design decisions

3 ALGORITHM DESIGN

4 USER INTERFACE DESIGN

5 REQUIREMENTS TRACEABILITY

6 EFFORT SPENT

Lorenzo Casalino

- 26 November 2016 - 1 hour
- 27 November 2016 - 3 hours and 20m
- 28 November 2016 - 1 hour and 10m
- 29 November 2016 - 1 hour
- 1 December 2016 - 2 hours
- 2 December 2016 - 1 hour
- 3 December 2016 - 2 hours
- 5 December 2016 - 1 hour and 30 minutes

7 REFERENCES