DD

Design Document
a.a. 2022/2023

Version: 1.0
Date: 08/01/23

Eutizi Claudio
Person ID: 10812073
Student ID: 995635
email: claudio.eutizi@mail.polimi.it

Perego Gabriele
Person ID: 10488414
Student ID: 987104
email: gabriele2.perego@mail.polimi.it

Contents

1 Introduction

1.1 Purpose

The purpose of the following **Design Document (DD)** is to deal with the architectural description of the *eMall* system and its services in order to provide more technical details about the project. While the previous RASD provided an abstract overview of the goals of eMall application, this document will focus on an in-depth description of the system's components, the interactions between them and their deployment. Then, it will be shown how the presented components will satisfy the Requirements already expressed in the RASD. This document will be also a guide for the implementation and testing phases.

1.2 Scope

As previously explained in the already published RASD, the *eMall* system wants to be a service that supports the charging process of electric vehicles, both for Drivers and for the Charging Point Operators. For this reason, the eMall system has to be able to satisfy needs of different kinds of Users and that is why its functionalities have been divided into two subsystems: *eMSP* and *CPMS*.

The first one is dedicated to provide the end-user services e.g. know the position of nearby charging stations, book a charge in a charging station, charge an electric vehicle and pay for the service etc.

The CPMS system administers the IT infrastructure of a CPO and provides a simple access point to the CPOs Operators that want to monitor the charging stations status, make decisions and apply changes e.g. handle the distribution of energy to the vehicles connected to a charging station, monitor the status of the charging station, decide from which DSO to acquire energy, set prices and offers etc.

In order to avoid the Operators to do too much repetitive operations, some of the listed operation can be also automatically performed by an autonomous system installed in the CPMS infrastructure. As already specified in the RASD, This autonomous system is sometimes mentioned, but its implementation and functionalities are not considered as part of the scope of these R&DD documents; it is, in fact, considered as an already existing and working subsystem that operates autonomously, but whose role is subordinate to the Operator's intervention, that is what this project is focusing on and has always an higher priority.

1.3 Definitions and abbreviations

1.3.1 Acronyms

- **RASD**: Requirement Analysis and Specification Document.
- **DD**: Design Document.
- **CPO**: Charging Point Operator.
- **eMSP**: e-Mobility Service Provider.

- **CPMS**: Charging Point Management System.
- **DSO**: Distribution System Operator.
- **QR Code**: Quick Response Code.
- **UML**: Unified Model Language.
- **API**: Application Programming Interface.
- **OS** : Operating System.
- **DBMS** : Data Base Management System.

1.3.2 Definitions

- **Client** : Software System on the user's device that requests services to the Server.
- **Server** : Software System that handles requests from different clients.
- **n-tier** : Distributed architecture composed of n hardware components, each containing one or many layers.
- **n-layer** : Distributed architecture composed of n software levels, each distributed on one or many tiers.
- **Design Pattern** : Reusable software solution to a commonly occurring problem within a given context of software design.
- **DBMS** : System Software for creating and managing databases.
- **Maps API** : Software platform that provides accurate real-time data in order to geolocate Users.
- **Business Logic Layer** : Layer which manages the services that eMSP and CPMS offer to Users (Drivers and Operators).

1.3.3 Abbreviations

- $[G.n]$ = n-th goal.
- $[R.n]$ = n-th functional requirements.
- $[D.n]$ = n-th domain assumption.
- $[UC.n]$ = n-th use case.

1.4 Revision history

Version	Date	Details
1.0	04/01/23	DD first draft

Table 1: revision history

1.5 Reference Documents

- R&DD Assignment AY 2022-2023
- RASD
- StarUML for diagrams

1.6 Document Structure

- **Introduction** (section 1): this section presents a general overview of the DD. It provides an overall description of the purpose and scope of the project and summarizes what will be presented in the following chapters.
- **Architectural Design** (section 2): this section details all the system level components. It presents a description of some of the architecture views by the mean of several UML diagrams. In particular the focus will be on the component, deployment and runtime view. The rationales behind the architectural styles and the design patterns adopted close this chapter.
- **User Interface Design** (section 3): this section is dedicated to the User Interfaces, but we have already included this part in the RASD, Section 3.1.
- **Requirements Traceability** (section 4): In this section each requirement of the RASD will be mapped with the design component that satisfies it.
- **Implementation, Integration and Test Plan** (section 5): this section aims to provide a plan that must be followed for the implementation of the components, their integration and the related testing.
- **Effort Spent** (section 6): this section shows a table that records the time spent by each group member for each section working on the DD part.

2 Architectural Design

2.1 Overview: High-level components and their interaction

The functionalities of eMall, as already mentioned in the RASD, will have different target devices with respect to its subsystems: a mobile application (Android, iOS) will provide Drivers the access to the eMSP functionalities, while both mobile and web applications will be developed for what concerns the CPMS system services. The eMall System will be designed as a distributed application, following the classic **three-tier architecture**. The three-tier architecture provides a subdivision of the application into three different layers, dedicated respectively to the *user interface*, *the functional logic* (business logic) and the *data management* (data logic). In this way, each of the three layers can be independently modified, giving scalability and maintainability to the application. Also, the decision to use a three-tier architecture was made in order to separate the business logic of the system from the data, which could be used for other applications in the future, and so decoupling them facilitates that. Below, there is first a short description of every tier, and then a graphical deployment view.

- **Presentation Tier**: this layer manages the interaction with the user taking care of the management of the external events. It performs simple tasks and invokes the logic tier when more complex operations are required.
- **Logic Tier**: it represents the core of the application. It contains all the components that constitutes the application logic.
- **Data Tier**: it is the layer in charge of managing all the data processed by the application.

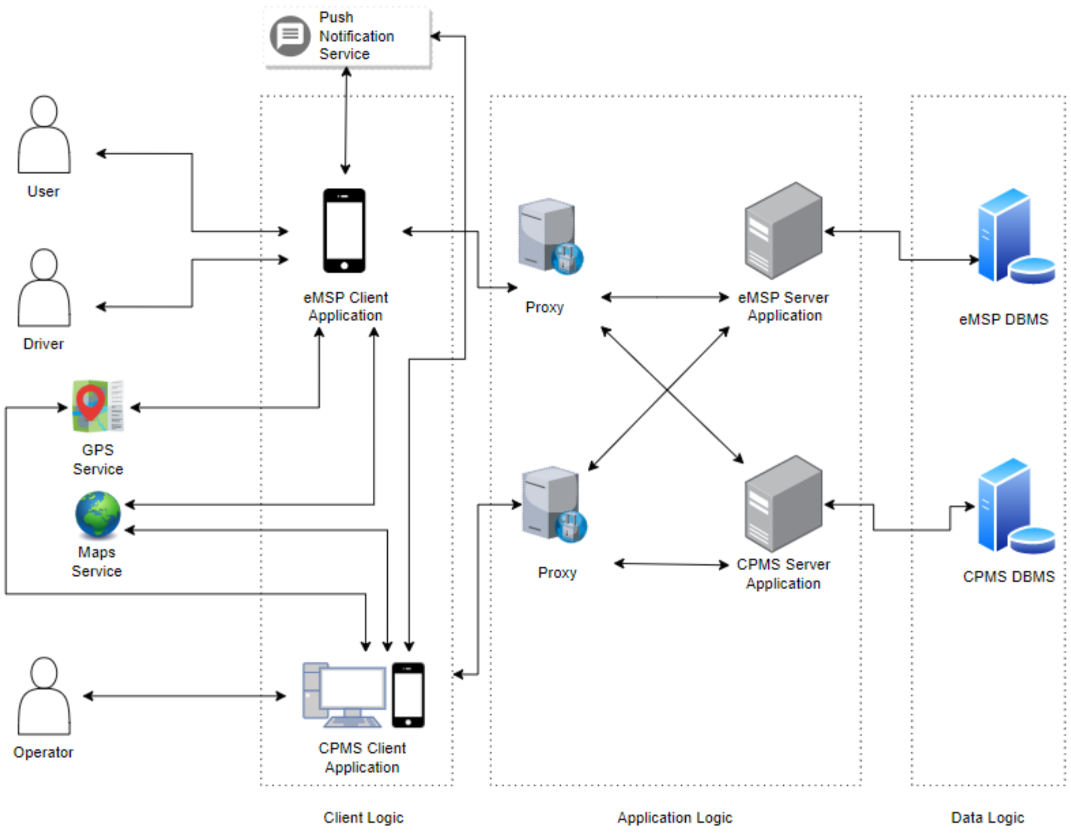

Figure 1: Overview of the System

The above image provides a general overview of the System's architecture.
We can immediately identify two horizontal layers and four vertical ones. The horizontal layers correspond to eMSP and CPMS architectures which have their Client Applications, Proxies, Application Servers and DBMS Systems. They also communicate to each other : eMSP accesses CPMS's functionalities and integrates them in its services by performing requests to its proxy. The vertical layers separate:

- The Client side, which includes Client Applications
- The Server side, which includes: Application Servers and the Proxies.
- External Database, accessed through its cloud interface.

Finally, also all external services are included in the diagram, both those accessed by the Clients, such as Maps, GPS and Health Monitoring services, and those accessed by the Servers, such as a Push Notification service.

2.2 Component view

2.3 Deployment view

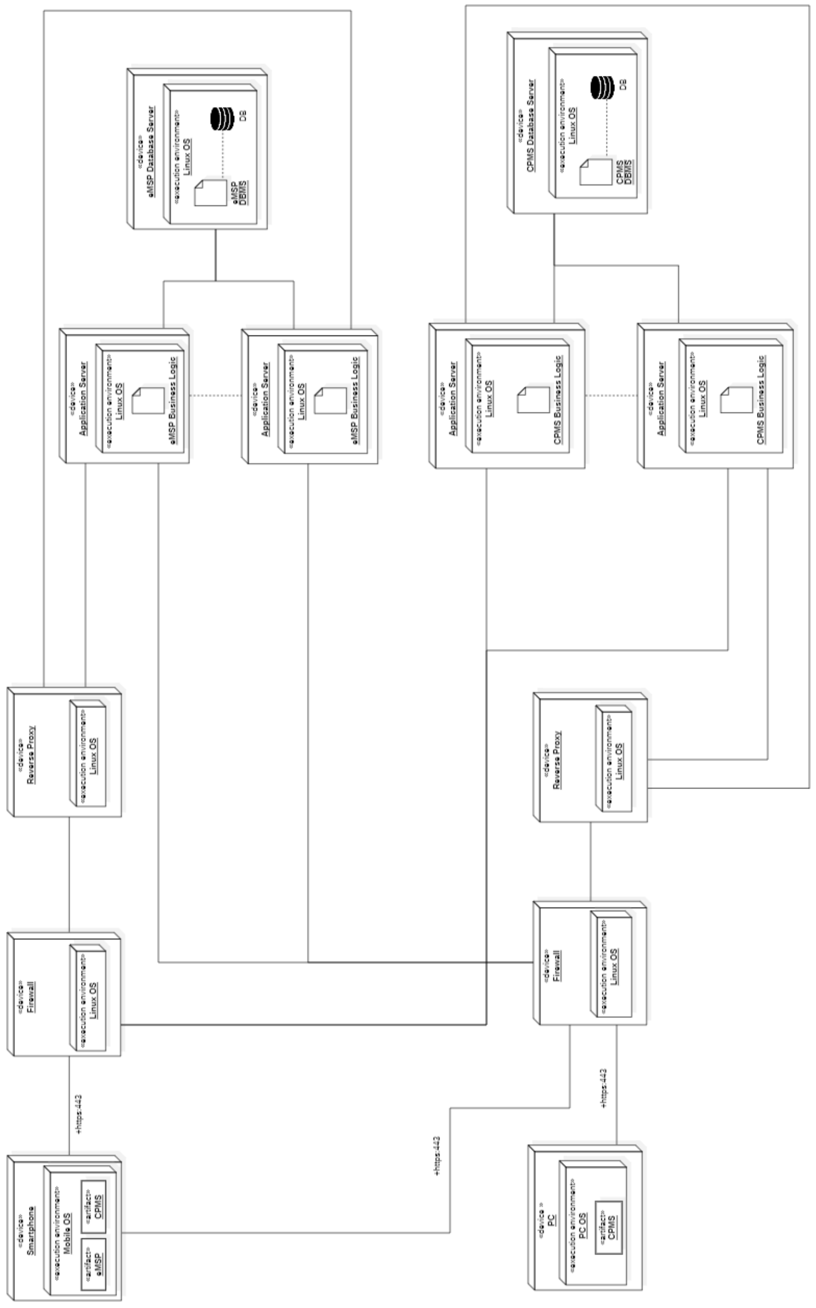
«device» eMSP Database Server
«execution environment» Linux OS
eMSP DBMS
DB
«device» Application Server
«execution environment» Linux OS
eMSP Business Logic
«device» Application Server
«execution environment» Linux OS
eMSP Business Logic
«device» CPMS Database Server
«execution environment» Linux OS
CPMS DBMS
DB
«device» Application Server
«execution environment» Linux OS
CPMS Business Logic
«device» Application Server
«execution environment» Linux OS
CPMS Business Logic
«device» Reverse Proxy
«execution environment» Linux OS
«device» Reverse Proxy
«execution environment» Linux OS
«device» Firewall
«execution environment» Linux OS
«device» Firewall
«execution environment» Linux OS
+https:443
+https:443
+https:443
«device» Smartphone
«execution environment» Mobile OS
«artifact» eMSP
«artifact» CPMS
«device» PC
«execution environment» PC OS
«artifact» CPMS

Figure 2: Deployment Diagram

The System (fig. 2) presents a multi-tier architecture in which the role of each node is specified below.

- **Mobile**: this node acts as a client machine and could hosts eMSP and CPMS applications.
  CPMS and eMSP applications run over a Mobile OS, in particular we consider the two most common ones that are AndroidOS and iOS.
- **PC** : this node works as well as a client and allows Operators to access to CPMS functionalities, through the use of a web browser.
- **Firewall**: this component filters the access to the Reverse Proxy and is used to protect a trusted network from an untrusted network. A firewall provides protection from unauthorized requests or from various types of malicious attacks.
- **Application Servers**: this level of the architecture encloses all the business logic of the Systems. CPMS and eMSP Application Servers are fully replicated to balance the workload. CPMS Application Servers communicate with eMSP ones (and viceversa) in order to make use of their APIs.
- **Reverse Proxy**: this node helps to achieve increased parallelism and scalability.
  It is a server that sits in front of a web servers, intercepting requests from clients. In the eMall System, the reverse proxy servers will intercept both client (Drivers and Users for eMSP and Operators for CPMSs) and server requests, because the eMSP and the CPMSs subsystems need to communicate with each other (e.g. the eMSP needs to retrieve information from a CPMS in order to keep up to date the presented data to the end-user). It is responsible for the load balancing as it distributes requests between all Application Servers. The Reverse Proxy also increases security and anonymity by protecting the identity of our back-end servers and acting as an additional shield against security attacks.
- **Database Server**: this machine is equipped with a relational DBMS and it is used to store and retrieve all data needed by the Application Servers.

2.4 Runtime view

2.5 Component interfaces

2.6 Selected architectural styles and patterns

2.7 Other design decisions

3 User Interface Design

User interfaces have already been described in section 3.1.1 of the RASD document where all mockups can be found.

4 Requirements Traceability

5 Implementation, Integration and Test Plan

6 Efforts

Individual Work		
	Eutizi Claudio	***Perego Gabriele***
Tasks	**Hours**	**Hours**
Introduction (section 1)	0	1
Architectural Design (section 2)	0	0
User Interface Design (section 3)	0	1
Requirements Traceability (section 4)	0	0
Implementation,Integration and Test Plan (section 5)	0	0
Final Revision	0	0
Total	**0**	**2**

Table 2: Time spent by each team member