



POLITECNICO
MILANO 1863

e-Mobility for All

Design Document (DD)

v1.0

A. I. Pascu (10703025)

G. Paolino (10696774)

L. Padalino (10695959)

Academic Year

2022/2023

Contents

1	Introduction	3
1.1	Purpose	3
1.2	Scope	4
1.3	Definitions, Acronyms, Abbreviations	4
1.3.1	Definition	4
1.3.2	Acronyms	4
1.3.3	Abbreviations	4
1.4	Revision History	4
1.5	Reference Documents	4
1.6	Document Structure	4
1.7	Overview	4
2	Architectural Design	4
2.1	Overview: High-Level Components and Interactions	4
2.2	Component View	4
2.3	Deployment View	5
2.4	Runtime View	5
2.5	Component Interfaces	5
2.6	Selected Architectural Styles and Patterns	5
2.6.1	Model-View-Controller (MVC) Pattern	5
2.6.2	Observer Pattern	5
2.6.3	Three-tier Architecture	6
2.7	Other Design Decisions	6
3	User Interface Design	6
3.1	User Mobile Application	7
3.1.1	Registration	7
3.1.2	LogIn	8
3.1.3	Home Page	9
3.1.4	Charging Station Details	10
3.1.5	Booking	11
3.1.6	Trip Planning	12
3.1.7	Charging Process	13
3.1.8	Payment	14
3.2	CPO Client Application	15
3.2.1	Special Offer Proposal	15
4	Requirements Traceability	16
5	Implementation, Integration and Test Plan	16
5.1	Implementation	16
5.2	Integration and Test Plan	17
6	Effort Spent	18

1 Introduction

1.1 Purpose

The main challenge of a massive deployment of electric mobility is the reduction of transportation's impact on climate by limiting the carbon footprint caused by our urban and sub-urban mobility everyday needs.

With this in mind, our aim is to develop and implement a new system called eMall having the purpose to expand the electric charging infrastructure with the means of realizing a world-wide network which fully connects all the actors involved in the charging processes. This is achieved by making all the back-end operations and transactions transparent to the everyday user which is fully supported in the charging process.

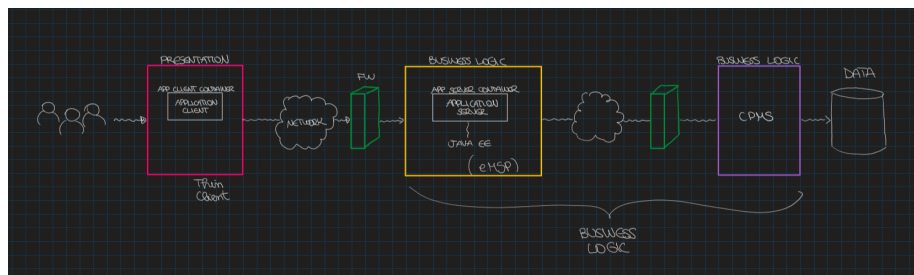
eMall will act as a platform deployed as a mobile application and implemented through several servers (eMSPs) located throughout the globe allowing to monitor electric mobility by communicating with various CPOs, each of them administrated through a CPMS.

Thereby the platform will expose several services such as:

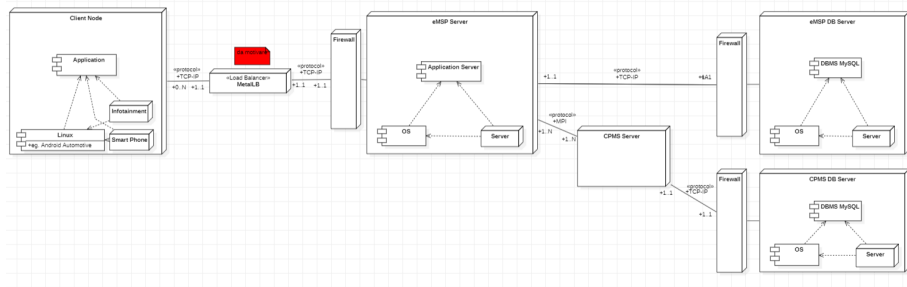
- knowing where to charge the e-vehicle (locate charging stations owned and managed by CPOs)
- planning charging processes in a way to limit constraints on our daily schedule giving the possibility to the system to access our personal calendar
- choose from various charging possibilities based on special offers set by CPOs which can dynamically decide from where to acquire energy to be distributed (from which DSO)

In order to guide the development of the system to be step by step, the document will focus on the description of the architectural design for the system's components alongside with their interaction. Finally, several mock-ups of the user interface are displayed followed by a guideline regarding testing and integration.

2.1 Overview: High-Level Components and Interactions

[illegible]

2.3 Deployment View



2.4 Runtime View

Suggestion nel pdf dell'assignment: possiamo usare dei sequence diagrams per descrivere il modo con cui i componenti interagiscono per realizzare alcuni tasks tipicamente collegati agli use cases.

2.5 Component Interfaces

2.6 Selected Architectural Styles and Patterns

2.6.1 Model-View-Controller (MVC) Pattern

In project implementation, we wanted to use the Model View Controller (MVC) pattern, which allows for a modular and easily scalable application. In detail, the Model plays the role of a container of information, appropriately taken from the database, implementing also the logics to dialogue with the business layer of the application; the Controller has the task of interpreting the actors' requests and instantiates and enhances the appropriate model and, subsequently, of routing the request to the right components; the view is the template according to which the model is represented: it does not contain any application logic and is only and exclusively in charge of displaying the data coming from the model. In our project, it was implemented client-side using frameworks in order to optimize resources.

2.6.2 Observer Pattern

To handle event notification in the crucial phases of vehicle charging rather than in the phases of event notification to users, we devised the Observer pattern. This is a software design pattern in which an object, called a "subject," maintains a list of its dependencies, called "observers," and automatically notifies

them of any state changes, usually by calling one of their methods. It is often used to implement distributed event management systems in event-driven software.

2.6.3 Three-tier Architecture

The system is structured into three layers: a presentation layer, a business logic layer, and a data layer. The choice of this architecture was mainly for reasons of security, reusability and scalability of the system. The data needs greater protection and should be separated from the application logic both conceptually and physically, defended by very restrictive security policies and at the same time able to allow multiple accesses quickly. The other layers have been separated in order to ensure possible node replication (load balancing) and possible resource replacements and upgrades, so as not to impact the other nodes in the architecture.

2.7 Other Design Decisions

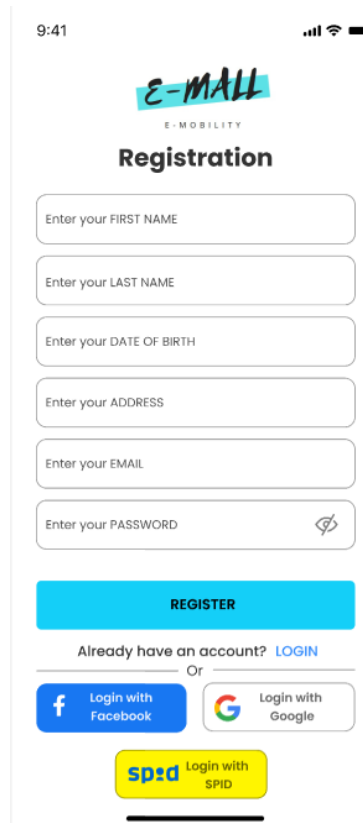
As can be seen in the component interface, we thought of separating the access points to the system to ensure different priority and security for requests coming from CPOs/CPMSs and those coming from users. Each interface has a different level of protection, particularly the one that allows the application tier to talk to the databases via the DBMS. In addition, the system makes use of external APIs with which to perform verifications and associations of input data.

3 User Interface Design

Following are some prototypes of the application user interfaces that users use to register and access the portal, search, book, and monitor the status of their EV charges.

3.1 User Mobile Application

3.1.1 Registration



The image shows a mobile application registration form for 'eMall'. At the top, the status bar shows the time 9:41 and signal icons. The app logo 'eMALL' is displayed in a stylized font, with 'E-MOBILITY' written below it. The title 'Registration' is centered. The form consists of six input fields: 'Enter your FIRST NAME', 'Enter your LAST NAME', 'Enter your DATE OF BIRTH', 'Enter your ADDRESS', 'Enter your EMAIL', and 'Enter your PASSWORD'. The password field has an eye icon for toggling visibility. Below the fields is a blue 'REGISTER' button. Underneath, there is a link 'Already have an account? LOGIN'. A horizontal line with 'Or' in the center separates the login link from the social login options. There are three buttons for social login: 'Login with Facebook' (blue with Facebook logo), 'Login with Google' (white with Google logo), and 'Login with SPID' (yellow with SPID logo). The SPID logo is stylized with 'spid' in blue and 'SPID' in black.

Figure 1: Registration Form UI

In this mock-up user has to fill the form in order to register to eMall. It's possible to use external provider such as Google or Facebook or use digital identity provider such as SPID. In order to perform the association between user and EV, the app asks the user to provide EV details.


3.1.2 LogIn

9:41

E-MALL
E - M O B I L I T Y

Login
Welcome back!

Enter your EMAIL


Enter your PASSWORD 


[Forgot Password?](#)

LOGIN

Don't have an account? [REGISTER](#)

Or

 Login with Facebook

 Login with Google


 Login with SPID

Figure 2: Login Form UI

To access to all app features user must be logged by filling this form or using external provider used in the registration phase.

3.1.3 Home Page

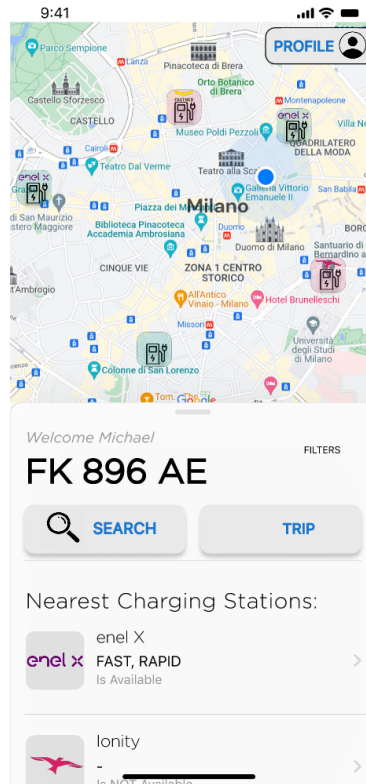


Figure 3: Home UI

After have been logged into eMall, the home page load user position and EV details, then displays nearby charging stations, adding CPO and charging point availability.

The user can use this UI also if he wants to look for a charging station by using custom filters or to plan a trip.

3.1.4 Charging Station Details

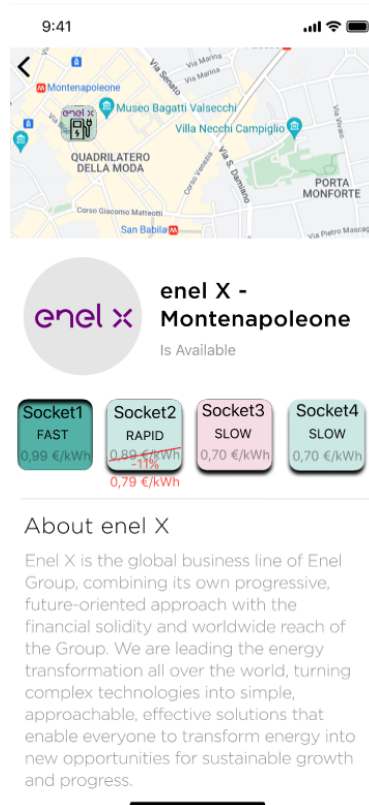


Figure 4: Charging Station UI Details

This is the UI displayed after a charging station lookup containing all the available information in terms of sockets, prices, special offers, status.

3.1.5 Booking

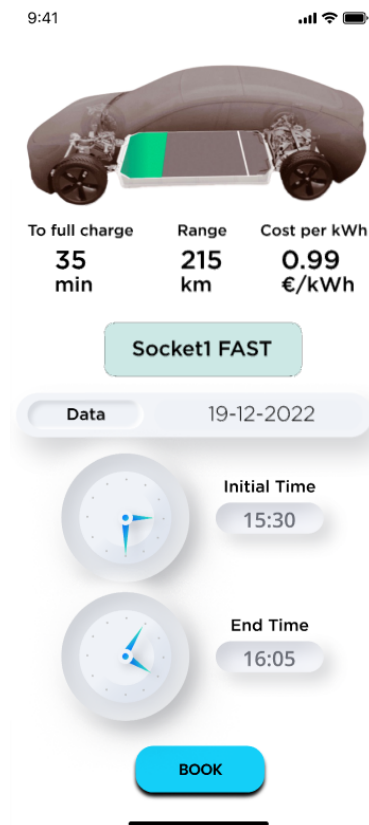


Figure 5: Charging Station Booking UI

The user can access to this UI after having clicked on the socket in the previous screen. After having inserted date and time of charge, he can book the charging station and so save the task in his calendar (e.g. available if the user has associated his Google account) if and only if there aren't event overlaps.

3.1.6 Trip Planning

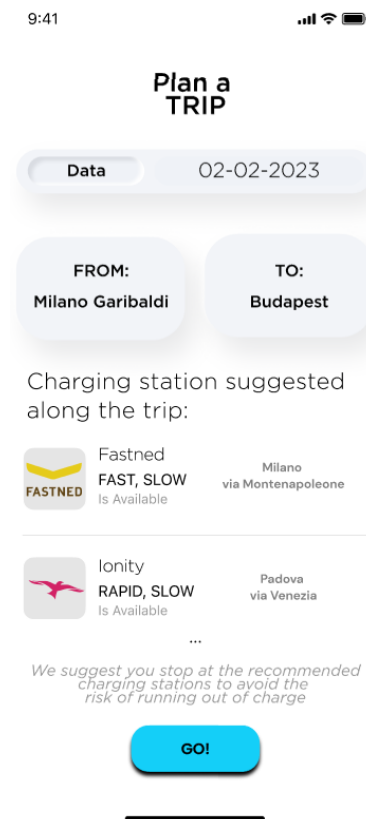


Figure 6: Trip Planning UI

If the user wants to plan a trip he can access all the available charging stations along his/her travel path. The charging stations are displayed according to EV's battery level with the purpose to optimize time spent on charging operation.

3.1.7 Charging Process

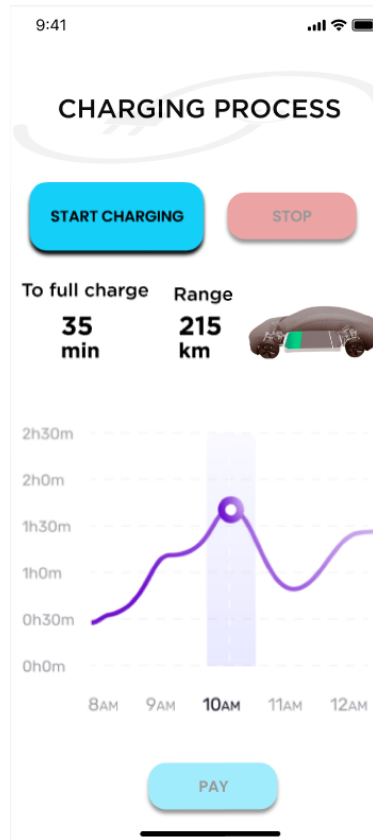


Figure 7: Charging Process UI

This is the UI that user must deal with to start a charge and manage all possible events which can occur during charging process. The user can access to this screen after having received a notification about having connected his/her EV to the socket. Once the process started, he/she can decide to stop and to pay only for the supplied energy, thus interrupting the charging process.

3.1.8 Payment

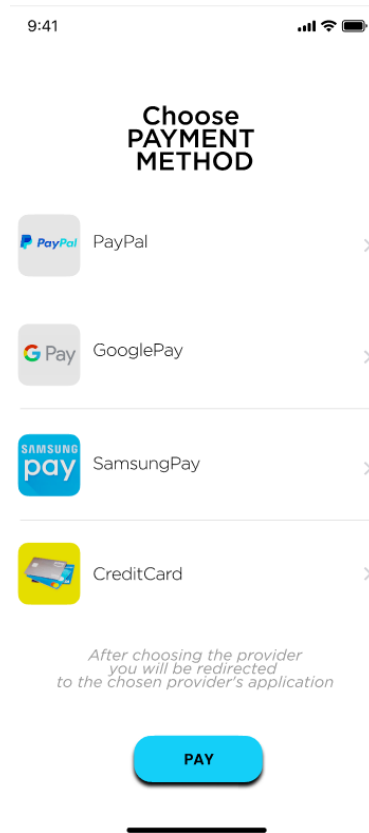


Figure 8: Payment UI

In this UI user must choose an external payment provider to complete the charging process. The transaction will be managed by the external provider, and the result will be transmitted to eMall. Now the user can resume his/her journey.

3.2 CPO Client Application

3.2.1 Special Offer Proposal

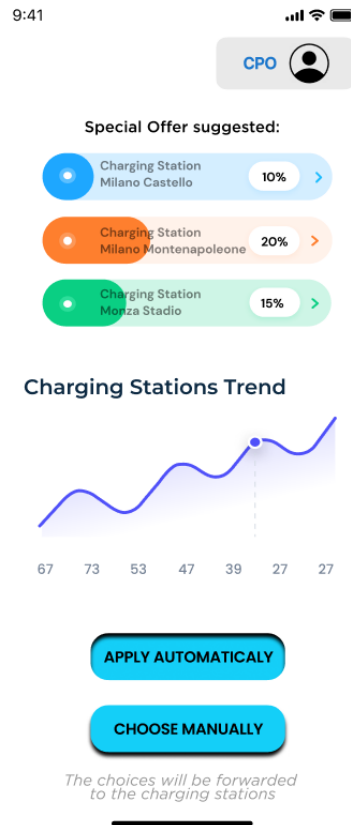


Figure 9: Special Offer UI

After CPO has logged onto the application, it can deal with all its charging stations, and it can also monitor their performances. Above there is the screen for making special offers based on charging station performance.

4 Requirements Traceability

Spiega come i requisiti che hai definito nel rasd sono mappati agli elementi di design che hai definito in questo documento

5 Implementation, Integration and Test Plan

This section focuses on the description of how the implementation, integration and testing of the system is to be applied.

5.1 Implementation

The implementation of the eMall system is doomed to follow a bottom-up strategy as far as implementation of components is concerned. Individual components of the system (included into the eMSP) are first specified in great detail. In particular, we divide the implementation into sub-groups in order to breakdown the whole work and maintain a clear organization of the development to facilitate the resolution of problems in case they may occur in the meantime. This approach also allows the monitoring and controlling of milestones and the associated deliverables in such as way to check the progress and decrease the possibility of failure of the whole project.

The Component Diagram plays a key role in dividing the tasks into the so called Work Packs (WP) to associate to the groups involved into developing the project.

- **Client Application**

View part of eMall system only concerning with the Client's interaction. This component is only responsible for the GUI and allows the Client to interact with the system and the features it exposes.

- **Model, eMSP DBMS, Data Visualization**

This components have to do with the data model. We use a Relational Data Base Model in order to facilitate the storage of data by using tuples. The ORM is easily achieved thanks to Java EE. Data Visualization is used for marshalling and serialization which both allow a better usage of data to maximize the storage space and also the transferring of data.

- **Request Handler**

This component receives various requests from the Client Application and forwards them based on the type of request to the correct Service. This could be achieved in terms of COTS or by implementing the Handler from scratch.

- **Services, Controller**

All the Components which provide services can be implemented in parallel due to their independent feature exception made for the Authentication and Authorization. This last two services secure the correct access to the

system and so to other Services; for this reason, they should be implemented first. Lastly, the Controller redirects the correct request to the CPMS and communicates with the Model in order to store information into the local DB.

- **Handlers**

Handlers are components that allow the communication with external sources; they can be implemented from scratch or acquired as COTS.

- **PerformanceCalculator**

This is a special component that needs to be implemented having in mind the SpecialOfferService as a stub. It periodically gets raw data from the eMSP DB; this data is being cleaned, and elaborated to extract useful analytics. The result is asynchronously sent to the SpecialOfferService which performs Machine Learning algorithms to find out the best way to improve the services provided by a specific CPO and to maximize its profit based on certain parameters.

Once the eMSP sub-system is fully operational, the integration with the CPMS will start.

5.2 Integration and Test Plan

The integration of the deliverables of each milestone such the completion of the implementation of one single component starts after its functionalities have been fully tested in isolation, which requires the presence of drivers. Drivers are needed as some components cannot work only in isolation so we need to simulate components.

Additionally, each component will be integrated following the bottom-up incrementally approach which facilitates bug tracking: for example, two components, tested in isolation and fully operational are to be merged together in order to achieve other functionalities of the sub-system. If the Oracle, which checks the test and the execution of the sub-system composed of the two components, assesses a correct state of the system and the expected output corresponds to the actual one, then the integration may proceed until the whole eMSP sub-system is fully integrated and operational.

Finally, the whole eMSP sub-system needs to be tested with the CPMS sub-system in order to verify the correct behaviour of eMall. At first, we use drivers that simulate the presence of CPMSs but the integration test needs to be run again to avoid inconsistencies and to be sure that everything works correctly.

Below a graphical representation on how to perform the integration test properly.

6 Effort Spent

7 References

References

- [1] European Parliament and Council of the European Union *Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation)* OJ L 119, 4.5.2016, p. 1–88
- [2] Diego Ongaro and John Ousterhou *In Search of an Understandable Consensus Algorithm*, Stanford University
- [3] Alex Kopestinsky *Electric Car Statistics In The US And Abroad*
- [4] Richard Denning *Applied R&M Manual for Defence Systems*, Chapter 6: Probabilistic R&M Parameters and Redundancy Calculations