Politecnico di Milano

Software engineering 2

PowerEnjoy
Design Document v1.0

Alessandro Caprarelli	874206
Roberta Iero	873513
Giorgio De Luca	875598

December 10, 2016

Contents

Chapter 1

Introduction

1.1 Purpose

This document represents the Design Document *DD*. The main purpose is to describe the system in terms of its components, providing a detailed description of the high level architecture, the design patterns we are going to use, the analysis of the internal and external interactions of the components. This document is addressed to developers that have to implement the software.

1.2 Scope

The aim of the project PowerEnJoy is to provide an automated service of car sharing. After a registration, a client can hire a car near him/her through the web or mobile application and he/she can enjoy of all the extra services offered. The exact position of the client is determined by the GPS signal of the client's device or it's allowed to manually insert a specific address. So the system displays all the available cars in the client's close area. Then the client could make a reservation of a car, after which the system notifies the client with a message of confirmation with the car identifier. If the reservation procedure successfully ends the chosen car won't be available anymore for other clients. Moreover a client cannot hire more than one car at the same time. After the reservation the client has at most one hour to reach the car, when this time expires the system gives a penalty to the client and the car, previously hired, is available again for other clients. The system allows the client to cancel his/her reservation. When the client reaches the car, he/she can tell the system that is nearby through a specific button in the application and he/she starts

to pay as soon as the engine ignites. During the travel the system supervises the current charge of the car and notifies it to the client through a screen located in the car. The system stops charging the amount of money that the client has to pay when he/she communicates through the application his/her decision to stop the rent. When the car is parked in a safe area and the client exits, the system locks the car automatically and starts the procedure of payment. The client is notified with the result of this procedure through an SMS, including the final fare.

1.3 Definition, acronyms, abbrevations

1.3.1 Definition

- **Guest client:** a person that is not already registered in the system or that has to log in.
- **Registered client:** a person who has valid access credentials to log in the system.
- **System administrator:** privileged user, in charge of managing administration processes and of updating business logic.
- **Reservation:** it is the action performed by a registered client that allow him/her to reserve an available car for maximum one hour.
- **Journey time = travel time:** time elapsed since the user starts the engine to the user parks the car and terminates the journey.
- **Available car:** a car that is not reserved by any user and has enough charge to be rented.
- **Unavailable car:** a car that is already reserved or damaged, so impossible to reserve.
- **Gps navigation:** it is the navigation system that is included in the car on board system. It could be used by the user to find direction to the final destination.
- **Final destination:** address where the user wants to go.
- **Safe area:** the region where is permitted to park and leave a car once the rent is terminated.
- **Power grid station:** the area where it's allowed users to park the cars, leaving them attached to the power grid.

1.3.2 Acronyms

- **RASD:** Requirements Analysis and Specification Document
- **DD:** Design document
- **API:** Application Programming Interface
- **UI:** User Interface
- **MVC:** Model View Controller
- **URL:** Uniform Resource Locator
- **UXD:** User Experience Design
- **BCE:** Boundary Control Entity
- **SMS:** Short Message Service
- **WSS:** Web Socket Secure
- **HTTPS:** HyperText Transfer Protocol over Secure Socket Layer

1.3.3 Abbrevetations

- **[Gn]:** n-goal
- **[Rn]:** n-functional requirement

1.4 Reference documents

- PowerEnjoy specification document (assignment).
- RASD v1.1 .
- IEEE Std 1016tm-2009 Standard for Information Technology - System Design - Software Design Descriptions.

1.5 Document overview

The document is substantially divided in 5 sections:

- Introduction: it provides a general description of the content of the Design Document and some additional information in order to be coherent with the RASD document, previously produced.
- Architecture Design: in this section there are a detailed increasing description of the components of our application, of their architecture and of their internal and external interactions. This section also explains the main design and architectural choices.
- Algorithms Design: this section contains an explanation of the main algorithms that have to be implemented in the software. Pieces of pseudo-code are included in order to give a clear view of those algorithms.
- User Interface Design: this section includes mockups and user experience explained via UX and BCE diagrams.
- Requirements Traceability: this section relates the decisions taken in the RASD to the design components introduced in this document.

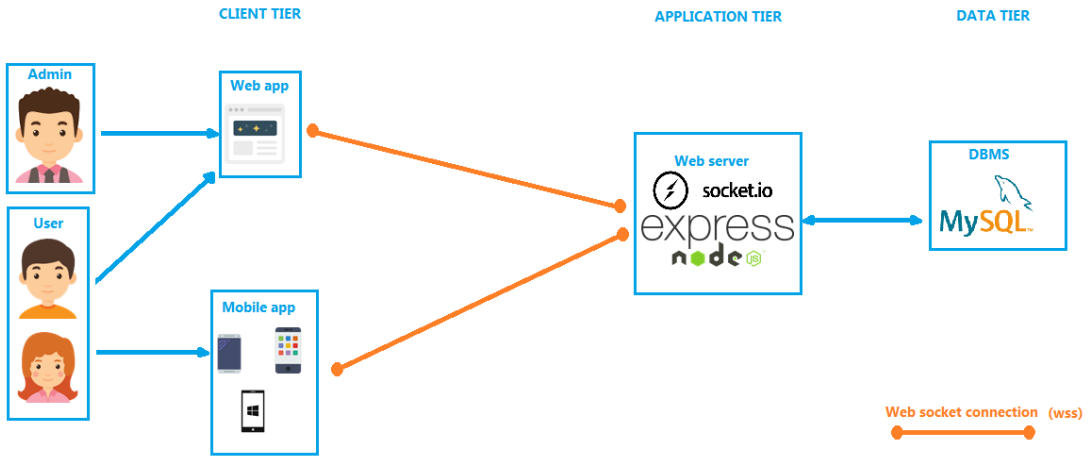
Chapter 2

Architectural design

2.1 Overview

This chapter describes the different components of PowerEnjoy and how they interact with each other, starting from an high level view of the system and finally explaining in detail the sub-components.

2.2 High level components and their interaction



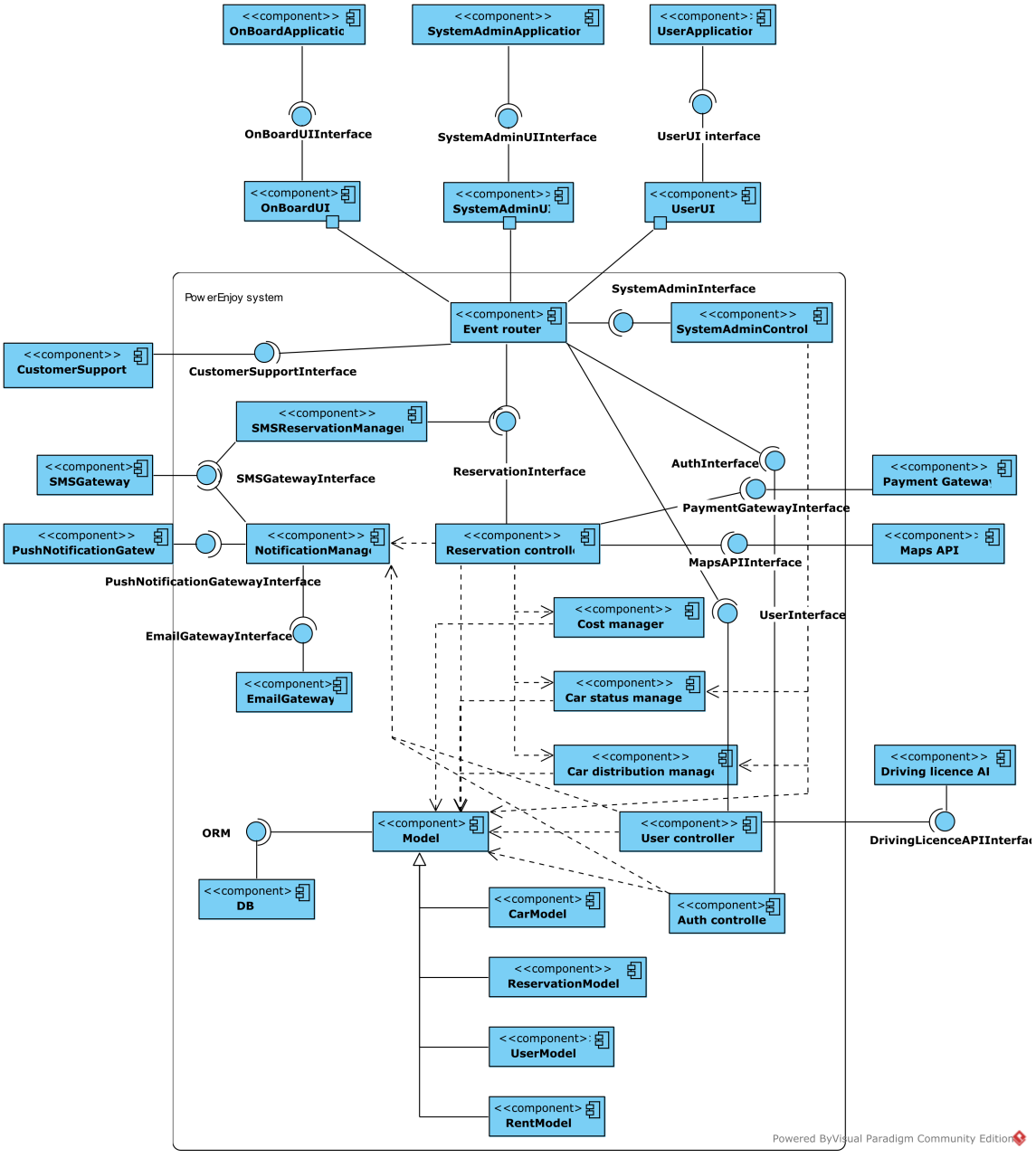
PowerEnjoy is composed by three main components: DBMS, web server and client application. These can be mapped to three tiers in a three-tiers architecture: the

data tier is the DBMS, the application tier is the web server and the GUI tier is the client application. The client application provides the UI through which the users can access the service offered by PowerEnjoy.

The UI makes requests to the web server that is in charge of elaborating those and eventually of providing a valid response. The client application is waiting for events from the web server and it updates the UI according to the data received.

The web server, while is processing the requests, can make call to external API and query the database.

2.3 Component view



The PowerEnjoy system runs a websocket server, as you can see from the high level view diagram, that receives events from the end-user applications.

The EventRouter dispatches the requests to the appropriate Controller that elaborates the request. During this operation the controller can use other components of

the system or can make calls to external services.

The ReservationController is the main controller and is in charge of managing all the process that involves a reservation, starting from the submission of a reservation request until the end of the rent. It can be called by either the EventRouter or the SMSReservationManager.

It uses some internal and external components in order to fulfill all the received requests:

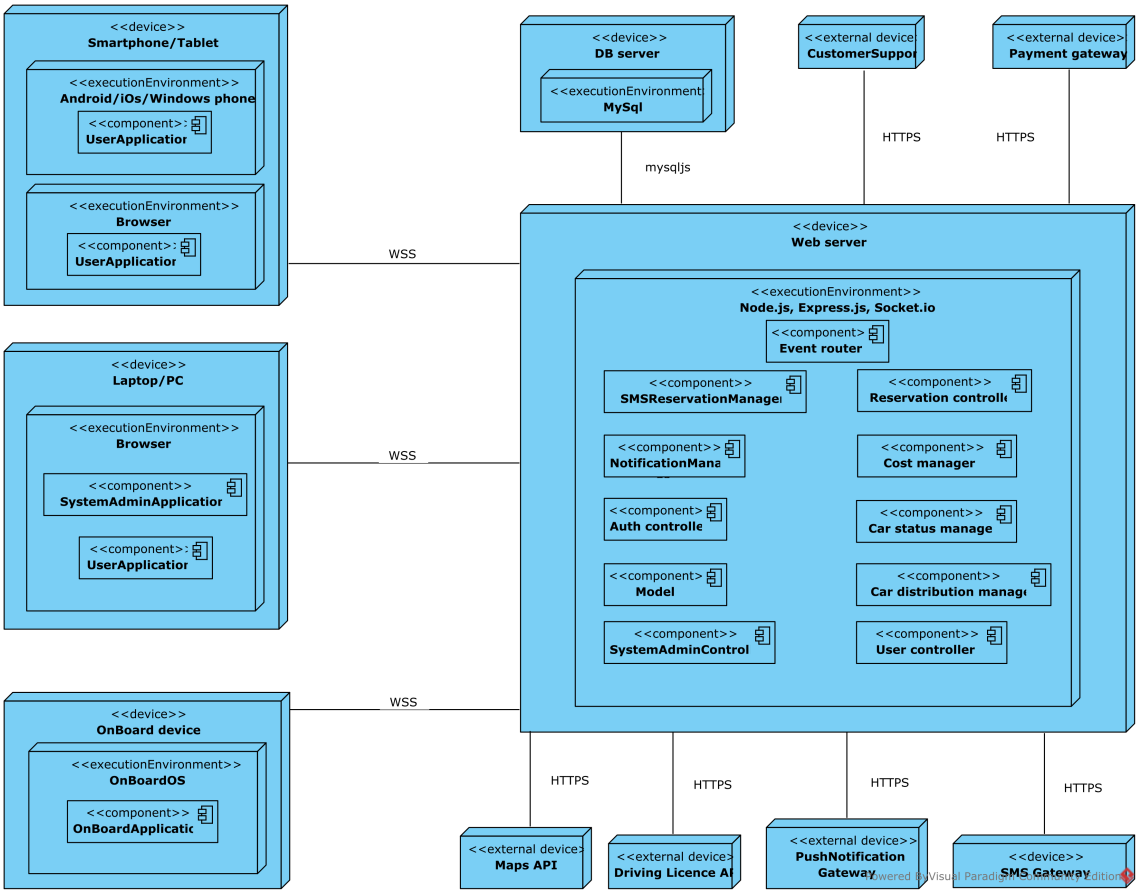
- it uses an external PushNotificationGateway in order to send real-time push notifications to users' mobile app.
- it uses an external SMSGateway in order to send SMS notification to users' mobile phone.
- it uses an external PaymentGateway in order to complete a payment after a rent is finished.
- it uses an external MapsAPI in order to locate users and cars into a map. In addition it uses the navigation API, that are part of the Google Maps API, in order to suggest the right journey to the user in case of 'save money' option.
- it uses the internal DiscountManager in order to calculate the discount to apply at the end of a rent.
- it uses the internal CarStatusManager and CarDistributionManager in order to check whether a car is available or not.

The AccountController is in charge of the process of registration and modification of user's profile and settings. It uses an external Driving licence API, provided by the 'Motorizzazione Civile', in order to check if a driving licence is valid or not.

The AuthController is in charge of checking the validity of the credentials during the login process.

All the controllers use instances of Model in order to retrieve data from the DB.

2.4 Deployment view



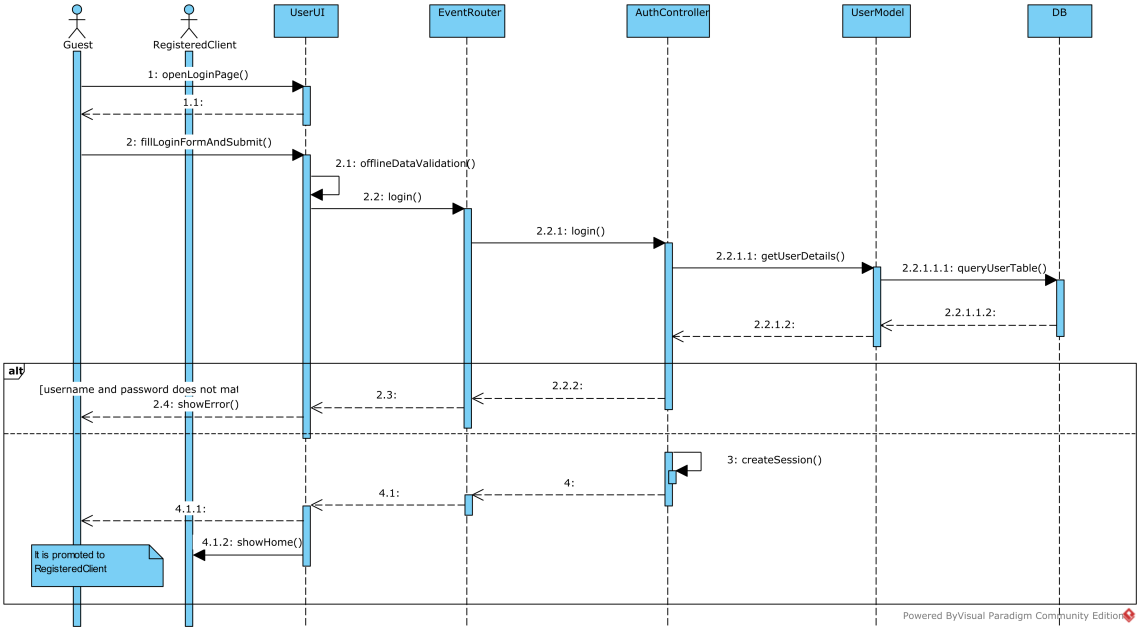
In this section the components defined in the previous section are placed within a device. The following diagram explains where each component is and how different devices communicate between them.

JSON is used to encapsulate data inside a standard structure and only JSON messages are exchanged between our internal components. This choice makes easier the exchange of messages, even complex objects, among components built with different technologies.

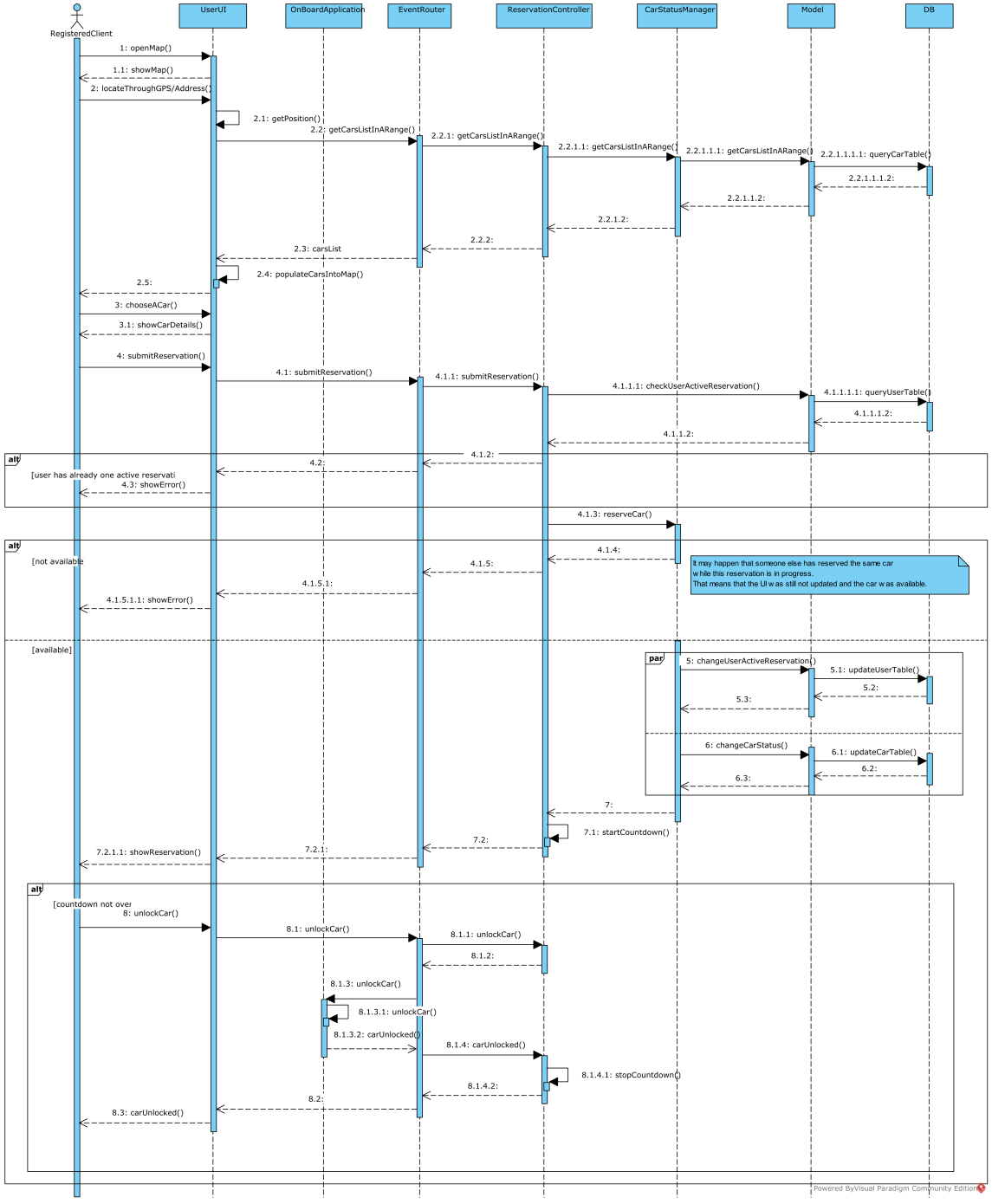
2.5 Runtime view

In this section are presented some sequence diagrams that explain the interactions among components showed in Component and Deployment diagrams.

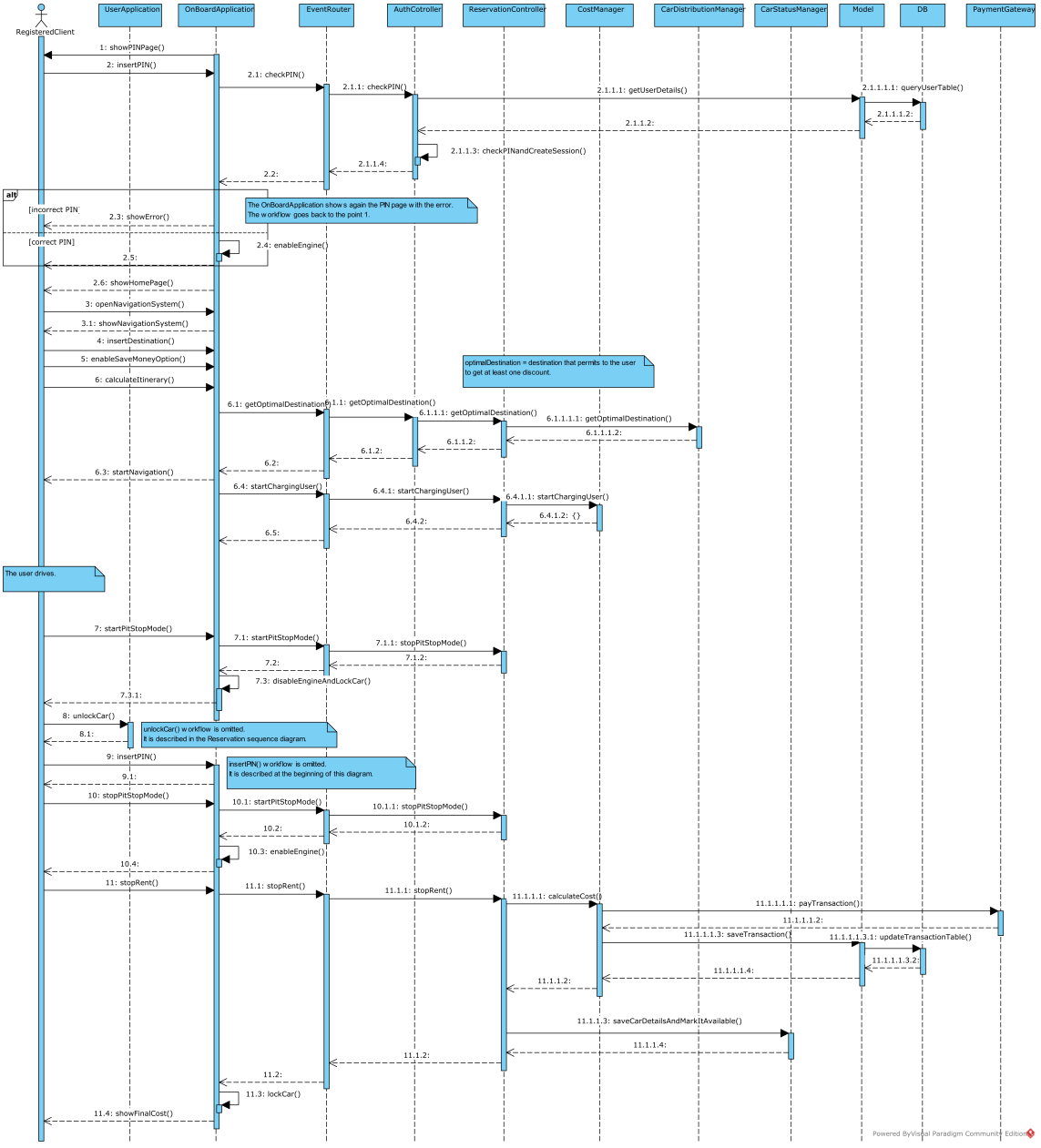
2.5.1 Login


Guest
RegisteredClient
UserUI
EventRouter
AuthController
UserModel
DB
1: openLoginPage()
1.1:
2: fillLoginFormAndSubmit()
2.1: offlineDataValidation()
2.2: login()
2.2.1: login()
2.2.1.1: getUserDetails()
2.2.1.1.1: queryUserTable()
2.2.1.1.2:
2.2.1.2:
alt
[username and password does not mat
2.4: showError()
2.3:
2.2.2:
3: createSession()
4:
4.1:
4.1.1:
4.1.2: showHome()
It is promoted to RegisteredClient
Powered ByVisual Paradigm Community Edition

2.5.2 Reservation through mobile app



2.5.3 Usage of on board application



2.6 Component interfaces

In this section all the interfaces, previously shown in the Component Diagram, are analyzed.

2.6.1 UserUI interface

This interface provides an entry point to the system for Guest users and Registered Clients, through web/mobile application.

The following list contains the main methods provided by this interface:

- showHome()
- showLogin()
- showSingUp()
- showAccount()
- showReservationPage()
- showMap()
- showCarDetails()
- submitRegistration()
- submitLogin()
- submitReservation()

2.6.2 SystemAdminUIInterface

This interface provides an entry point to the system for the System Administratror.

The following list contains the main methods provided by this interface:

- showHomeAdmin()
- showAdminLogin()
- showAdminAccount()
- showMenu()
- showDataEditor()

- submitAdminLogin()

2.6.3 OnBoardUIInterface

This interface provides an entry point for Guest users and Registered Clients, through the on board application.

The following list contains the main methods provided by this interface:

- showEnterPinScreen()
- showOnBoardHome()
- showNavigationSystemHome()
- submitAddress()
- submitSaveMoneyOption()
- submitTerminateRent()
- submitPitStopMode()
- stopPitStopMode()
- submitPin()
- submitAskForHelp()

2.7 Selected architectural styles and patterns

This section exposes the architectural decisions for the development the system of Power Enjoy and the reasons related to each choice.

2.7.1 Overall architecture

Service Oriented Architecture

SOA fits perfectly the needs of the system of Power Enjoy, starting from the calls to external services to the communication between the different parts of the data and business models, the involved devices and each architectural layer.

Client - server architecture

PowerEnJoy is an on-demand service usable by the client through website, smartphone app and SMS. The architecture that best complies with this context is the server-client one: besides all benefits of the communication between the devices, it's the most suitable architecture also for its maintainability and scalability. The interfacing with different types of devices is completely horizontal to the server side, permitting an easier implementation and management of the workflow of the system.

2.7.2 Design patterns

MVC

The Model-View-Controller paradigm is largely used in systems where a data model interacts with different clients, like PowerEnJoy: separating all components with respect to their aim permits to developers a better management of interactions between them.

SOTTOLINEARE CHE LA VIEW E' NEI CLIENT E MODEL E CONTROLLER IN SERVER

Factory

Useful to define interfaces able to encapsulate the information of a hierarchy of classes.

Singleton

Middleware

Chapter 3

Used tools and working hours

3.1 Used tools

The tools used to create this document are the following:

- MikTex 2.9: to format the document using LaTex.
- Pencilç to create mookups.
- Visual paradigm: to create sequence diagrams.
- LucidChart: to create use case, state chart and class diagrams.
- Alloy Analizer 4.2: to realize the model and to check its consistency

3.2 Working hours

	Alessandro	Roberta	Giorgio
28/10	1.5	1.5	1.5
29/10		0.5	
30/10	2		1
31/10	1.5	3	2.5
1/11	0.5	0.5	4
2/11	0.5	1	2.5
3/11	3	4	2
4/11	3	2	
5/11		3	3
6/11	1.5	3	
7/11	3.5	1.5	1.5
8/11	2	3	2
9/11	1.5		2
10/11	0.5		
11/11	3	3	3
12/11	3	3	3
13/11	1	1	1
Total	28	30	29