Final Presentation

Software Engineering 2 Project

Redaelli Marco Zanoli Francesco

Politecnico di Milano

February 28, 2017

Outline

Outline - Introduction

POLITECNICO MILANO 1863

Introduction

The project we have been assigned is called **PowerEnJoy** and it is a complex software system that should implement a car sharing service. In order to rationalize, clarify, and put in structured and standardized documents all the relevant concepts and informations, we designed and delivered several documents such as the **RASD**, the **DD**, the **ITPD,** and the **PPD**. These slides will only present an overview of the concepts thoroughly described in the above mentioned documents.

Tools

We composed the documents we had to using some tools such as:

- **TexStudio:** to compile LaTeX document.
- **StarUML :** to draw UML diagrams.
- **Alloy Analizer 4.2:** to checking model consistency.
- **Draw.io:** to build mockups and UML .
- **Skype:** for team collaboration.
- **GitHub:** for storing the project.

Outline - Requirement Analysis and Specification

Outline

Overview

The aim of the software is to provide a new digital management system for car-sharing service that exclusively uses electrical cars. It can be applied to different small and big city and even in a large urban area.

Goals

This new service pretends to achieve various goals, such as:

- **G5:** Allow user logged in to see the reservation's confirmation and the time of expiration
- **G6:** A user who request a rent can abort the process when ever he/she wants
- **G9:** A non registered users can only register once to the service.
- **G10:** A user can get discount or overtaxes from his/her last rent.

Actors - Users, Cars and Customer Service

Below are listed the four main actors that will interact with the application once deployed:

- **Registered User:** A person who is registered to the application system
- **Visitor:** A person who desires to subscribe to the system to benefit the full functionalities of the application.
- **Car:** The car reserved / rented from a User that communicates with the system.
- **External Customer Service:** A team that provides technical support in case of problems related to the car

Product Perspective

Our *PowerEnJoy* is a **completely new product**, not based on previous ones.

It relies on **location data** received via **Internet** from user application and the car system:all the involved smartphones already have a **GPS antenna** installed inside, that communicates their position to the service.

Being a partially **distributed application**, *PowerEnJoy* requires a fully operative **Internet** connection in order to work properly, both on server and client side: **no service is intended to be provided offline**.

More on Product Perspective

This software provides two **End User Interfaces**, a Mobile and of course a **Web Browser**, and a dedicated **Administrator** interface that is only accessible through a **LAN**.
All the data generated by this software are stored in a database, accordingly to current normative and laws about privacy and personal data management.
In addition, several **API**s are provided in order to allow further improvements and expansions of the software: for instance, the payment will be managed from an external entity (we suppose **PayPal**) and the map will be provided by **Google**.

POLITECNICO MILANO 1863

Outline

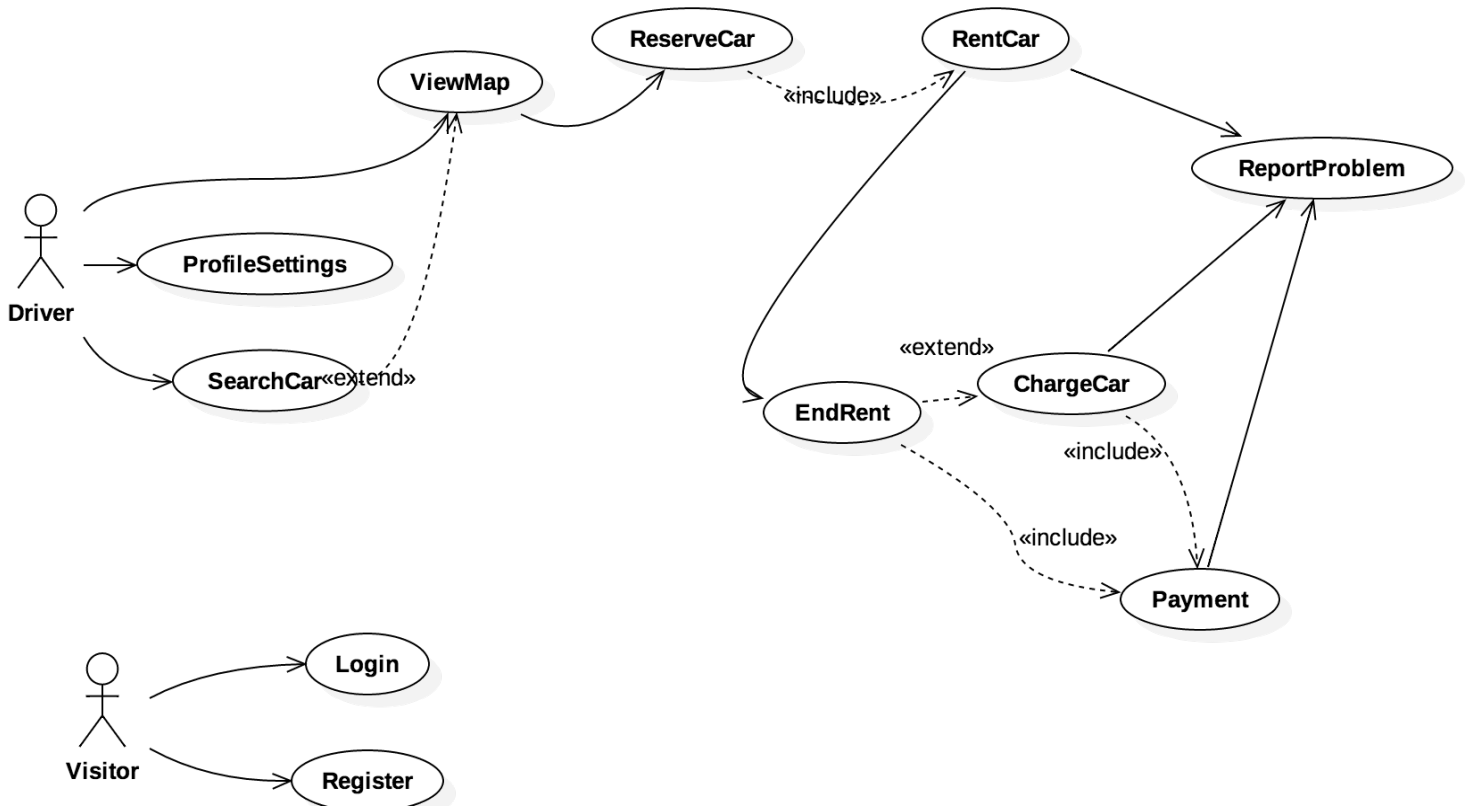
UML Diagrams

We provided a variety of UML diagrams, each type having a different purpose.

- **UML Use Case:** Shows the identified **use cases** in relation with the **involved actors**.
- **UML Sequence Diagram:** Indicates, for a given **use case**, the **interaction** between the **actors involved** and the **system**.
- **UML Class diagram:** Points out the different **software entities** involved in the application and the **relationships** between them.

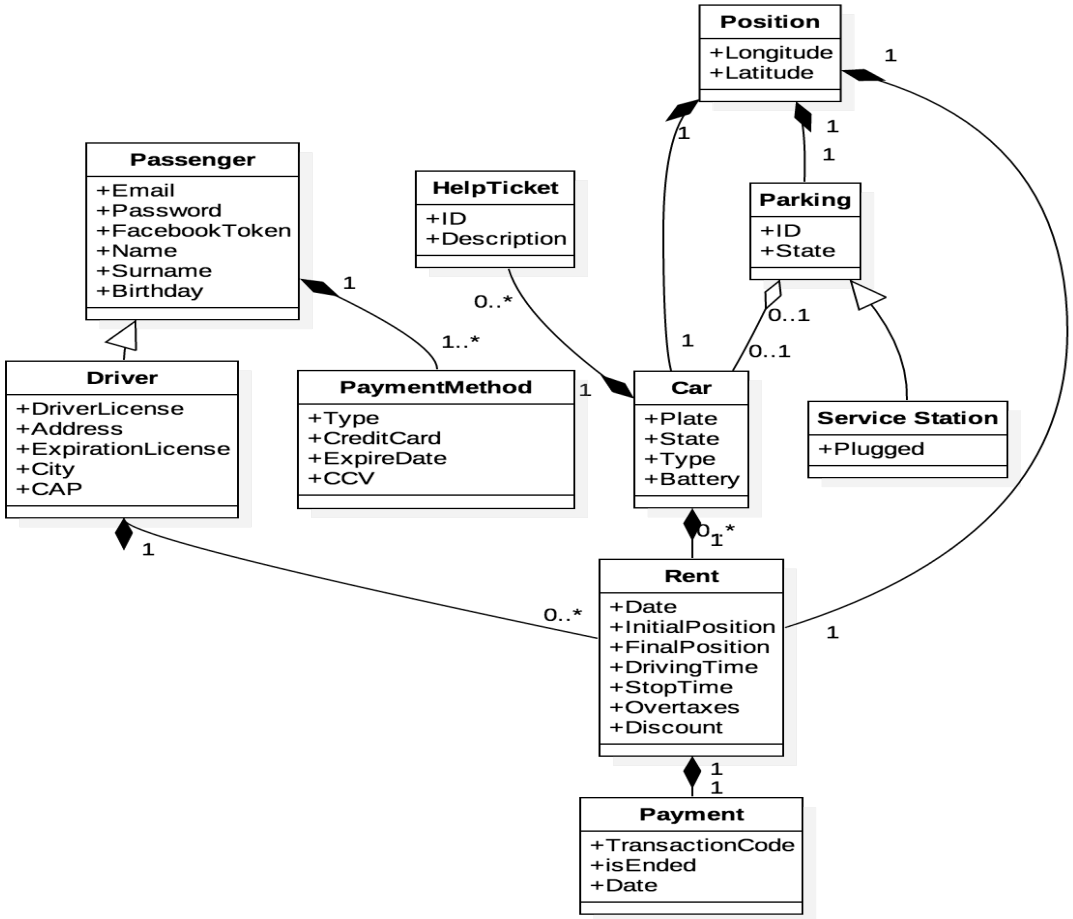
UML Use Case Diagram

This is perhaps the most useful diagram that can be designed in the early phase of the development of a software project.



UML Class Diagram

Furthermore we designed a class diagram for an early evaluation of the basic software components that consists in a sort of **Model** for *PowerEnJoy*.



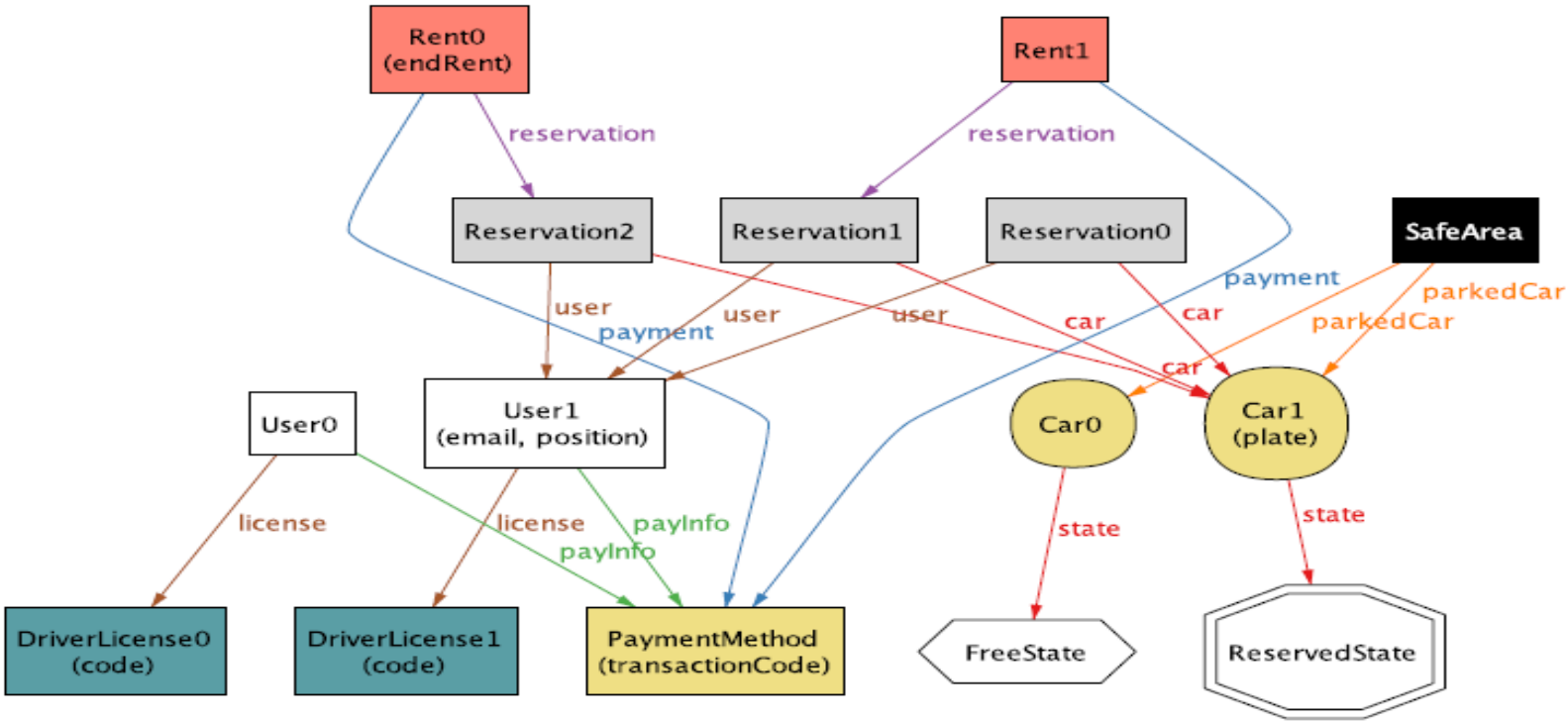
Outline

Alloy

Alongside the **UML Class Diagram** we built **Alloy Models** using the **Alloy** modeling language with the help of **Alloy Analyzer 4.2**.
The tool didn't find a proof of the inconsistency of our **Alloy Models**, and that along with the **Automatic Generation** (and **Manual Verification**) of interesting worlds, made us aware of the **Consistency** of those **Models** within a reasonable level of confidence.

Alloy Simple World

Here is an example of one among the **simplest world** we generated and double checked using both **Alloy Analyzer 4.2** and **manual checking**.



POLITECNICO
MILANO 1863

Outline - Design

Outline

POLITECNICO MILANO 1863

High level components and their interaction

The system is composed of many **distributed** components: those will communicate with a **Client-Server** style and through **Point to Point** messaging system.

- The **Client-Server** style is used to give the many Clients connected to the Server the opportunity of sending different requests (e.g. a **Car Reservation** or **Open Car Request**).
- The **Point to Point** bidirectional communication channel is made necessary to enable the Server the delivery of various messages and requests to the Clients:
  - Generic notifications
  - Service messages
  - The request of reserve a Car
  - The request of open a Car

Layers

The selected software architecture follows the principles of the **Model View Controller** architectural pattern, therefore three main software components have been identified and those are:

- The **Model**
- The **View**
- The **Controller**

Model, **View** and **Controller** are then mapped to three different relevant software layers.

Layers - View

This layer processes **Clients** commands, and converts them into requests addressed to the **Controller** layer. The **View** is connected to the **Controller** through a communication facility (e.g. The Internet).

Layers - Controller

This second Layer revolves around the concept of **Ride Manager**. Infact, we can say that is the core of the system because a very high number of requests are processed by this Manager. The controller sends and receives messages from both View side and Model side

Layers - Model

The third and last Layer is the **Model**. It:

- Guarantees a high level interface to store and manage all the *PowerEnJoy* relevant data.
- Abstracts a **Relational Database** in a software component that is in direct connection with the **Controller**

It has the responsibility of **receiving** and **handling** all the model updating

Tiers

The system is divided in **three** different tiers:

- **Clients:** The distributed clients of the application.
- **Application Server:** The most important Tier of the system. Here are done all the logics and calculations that constitute the core part of PowerEnJoy.
- **Database Server:** In this Tier it is hosted the Database that allows data persistence.

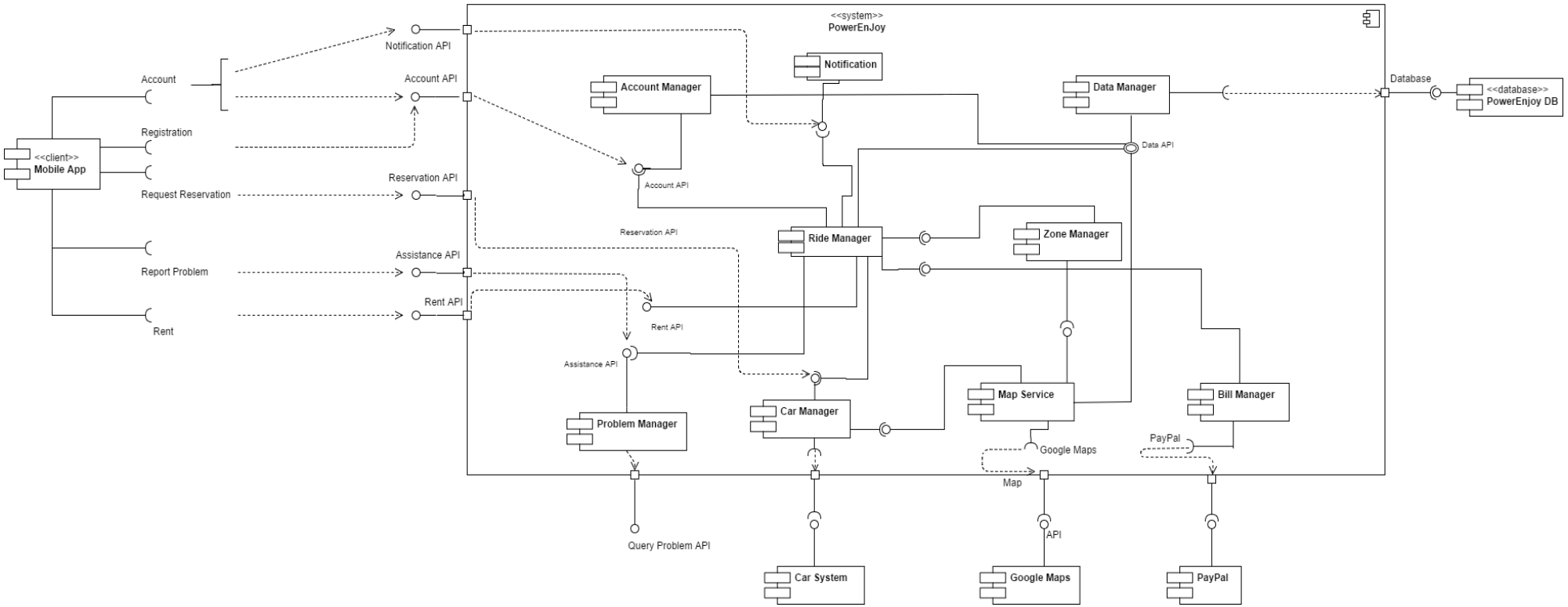
Component View

Several components has been designed to provide all the functionalities needed for *PowerEnJoy* to work. Many subsystems have been identified:

- Controller: Ride Manager, Bill Manager, Zone Manager...
- View
- Model

Component View - UML Component Diagram

This diagram maps system **features into different software components**, and show **how these components interact** in order to **deliver the required functionalities**. It helps showing **Layers organization** and the **MVC implementation**.



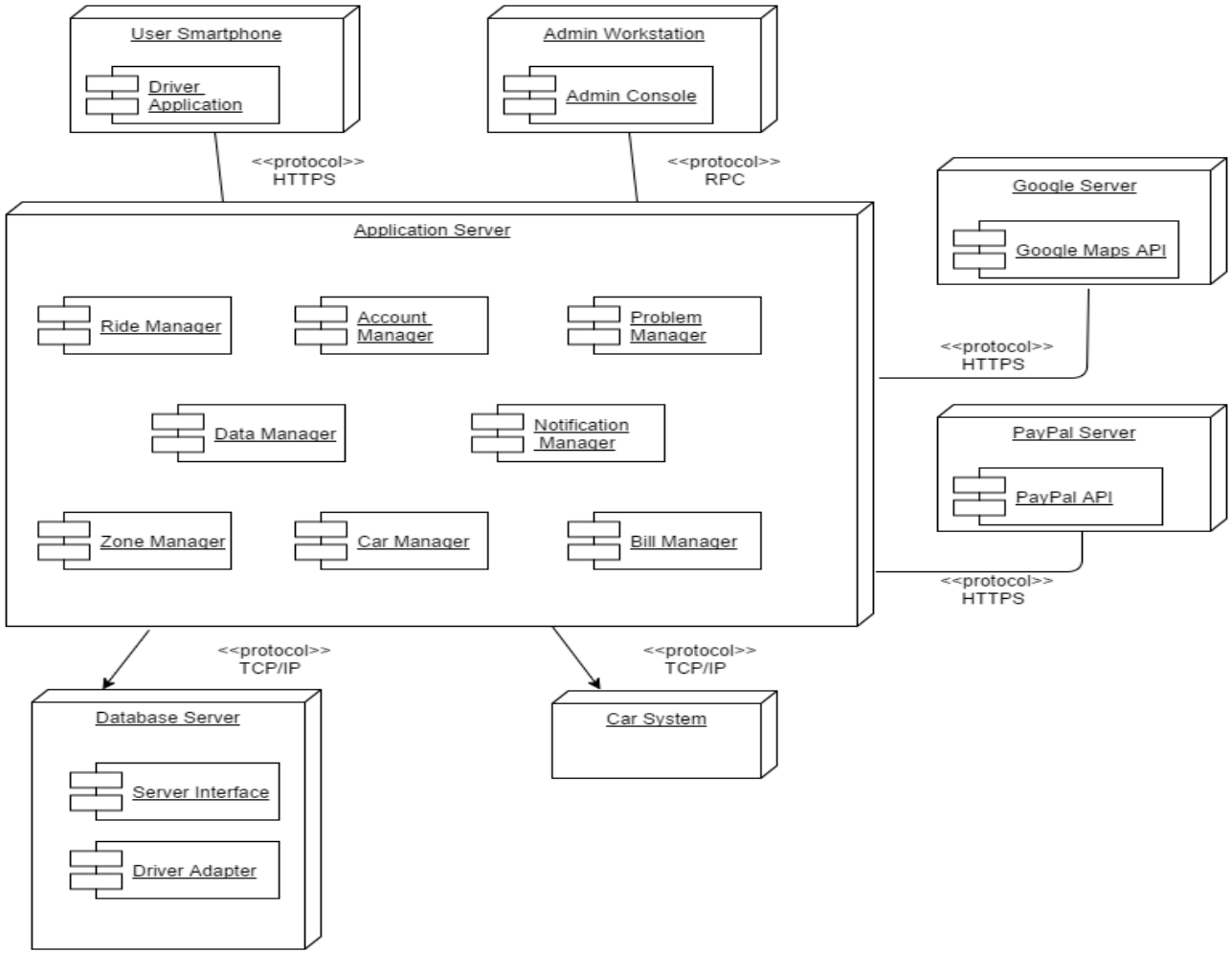
Deployment View

The best way found to **deploy** the software components identified, is to consider **5 different nodes** (7 if considering the Google Server and the PayPal server):

- User Smartphone, Administrator Workstation
- Google and PayPal Server
- Car System
- Application Server
- Database Server

Deployment View - UML Deployment Diagram

The following diagram shows how **software components** are mapped into the **physical system**.



Runtime View

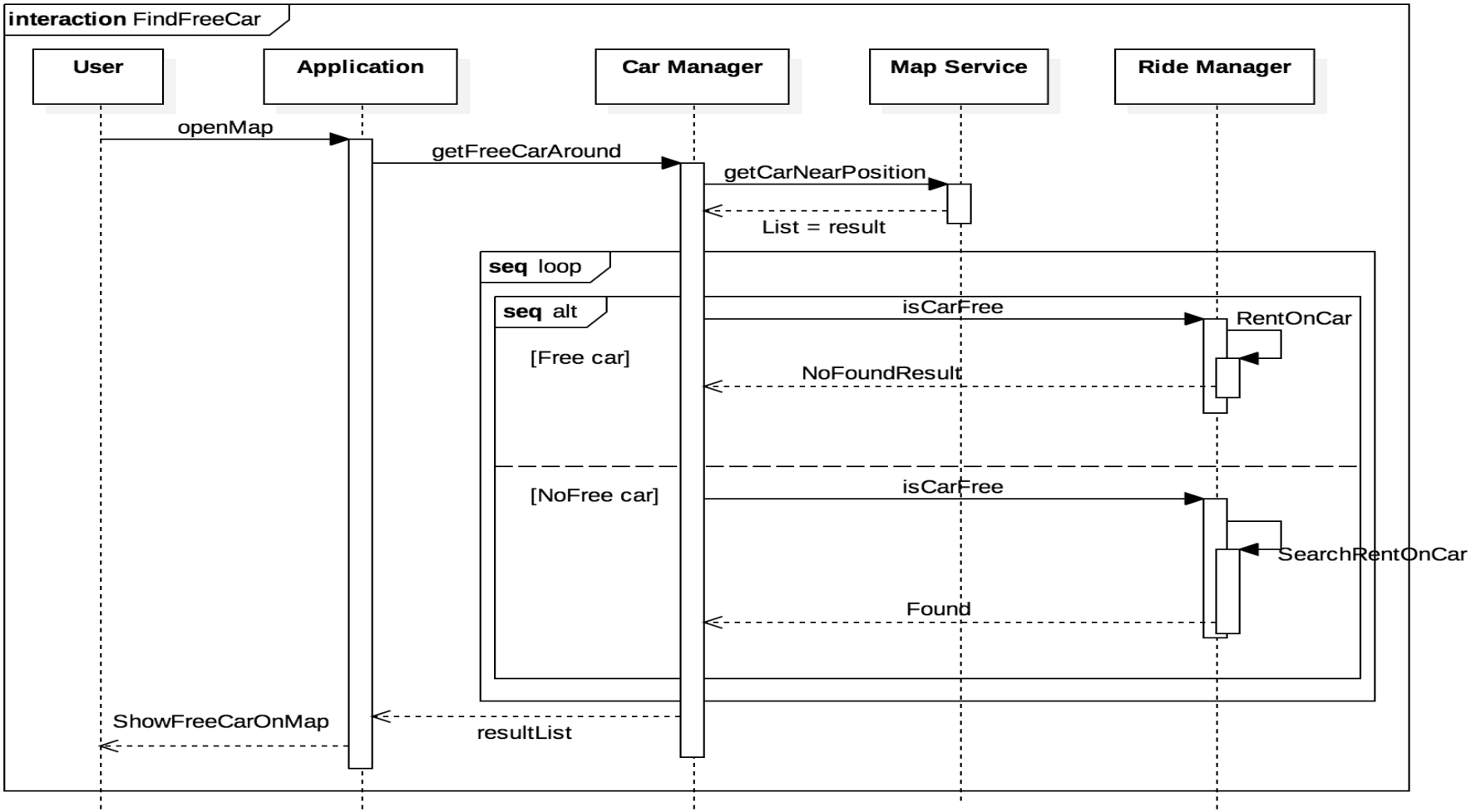
In this subsection are proposed some of the most meaningful **UML Sequence Diagrams** with respect to show how software components interacts in order to deliver a specific functionality. The chosen functionalities are:

- Find a free Car
- Make a Reservation

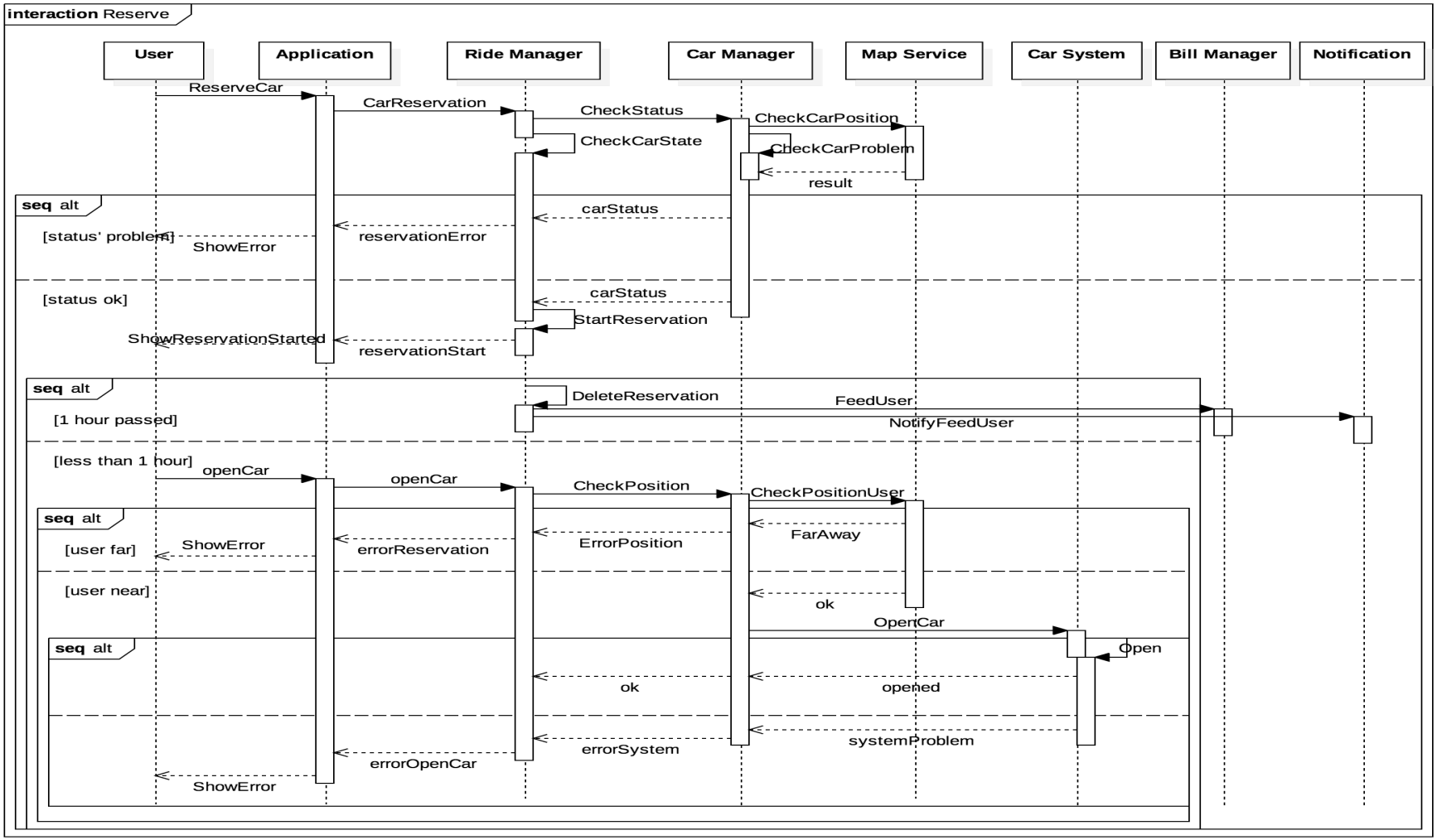
There are other functionalities whose **UML Sequence Diagram** is not reported here for space and time constraints:

- User Login
- User Registration
- Rent Car
- End Rent of the Car

Find free Car



Handling a Reservation



Selected architectural styles and patterns - MVC

Several architectural styles and patterns were chosen in order to build *PowerEnJoy* as a modern software. The main pattern that was recursively adopted is the **Model View Controller** architectural pattern:

- **System Level:** All the clients that use *PowerEnJoy* are seen as Views,that following the MVC pattern, are connected to a Controller,that through the Application Server is itself connected to the Model hosted on the Database Server.
- **Client Level**
- **Server Level**
  - Application Server
  - Database Server