

Politecnico di Milano
Software Engineering 2

Design Document
Bressan G., de Santis S., Di Marco P.

POWER ENJOY



Professor Elisabetta Di Nitto

Academic Year 2016-2017

Contents

Contents	1
1 Introduction	3
1.1 Purpose	3
1.2 Scope	3
2 Definition	4
3 Reference Document	4
4 Document Structure	5
5 Architectural Design	6
5.1 Overview	6
5.2 High Level Components and their Interaction	7
5.3 Component view	9
5.3.1 Database	10
5.3.2 Business Layer	11
5.3.3 Presentation Layer	12
5.4 Deployment view	13
5.5 RunTime View	14
5.5.1 Login	14
5.5.2 Close car	15
5.5.3 Open car	16
5.5.4 Reservation	17
5.5.5 Take a break	18
6 Component interfaces	19
6.1 UserManager	19
6.2 RideManager	20
6.3 DiscountManager	21
6.4 HistoryManager	22
6.5 ReservationManager	23
6.6 Vehicle Manager	24
7 Overall Architecture	25
7.1 Our Server to GM Server(External Server)	27
7.2 Design Patterns	28
7.3 Other Design Decision	28
8 Algorithm Design	29
8.1 Save money option	29
8.2 Discount algorithm	30
8.3 Open the car algorithm	31
9 User interface design	32
9.1 iPad System	32
9.2 Mobile App	33

10 Web App	36
11 Requirements Traceability	37
11.1 Functional requirements and components	37
12 RASD modification	38
13 Used tools	39
14 Hours of work	40
References	41

1 Introduction

1.1 Purpose

This is the Design Document for Power EnJoy App. It aims to provide an useful and complete view for the different stakeholders, in particular it is written for project managers, developers, testers and other people who are interested in different area of the business. In this part will be shown and described how Power EnJoy architectural components are integrated and their interaction with the algorithms. Moreover we will focus also on the RunTime behaviour of the system and we will present specific design patterns.

1.2 Scope

Customer has two different ways to interact with the system: Mobile App and Web App. In the first case, thanks to GPS system integrated in mobile phone the customer can choose, reserve and open the nearest car from him. In the second way the customer can only choose and reserve the nearest car from a selected point in a map. Once entered into a car, the customer has a direct interaction with iPad, in fact he have to insert a pin code, received on his mobile phone when he clicked on “open the car” button. After the correct validation of pin code, the system automatically recognise the customer, load all his information and the car is ready to go.

Power EnJoy System aims to offer a platform that allow customers to reserve shared electric vehicle using mobile app or web app. In according to provide an interaction between customers and Power EnJoy company in this document will be represented the different layers that compose the system to avoid a correct run time behaviour of the system. System also needs car's iPad to manage the communication with Power EnJoy server (using RESTful API) to take information about the OnStar hardware (that includes different type of sensors) installed in each car and that communicate with GM server using GM APIs. Moreover the system will include different extra functionalities such as bonuses according to the user behaviours.

2 Definition

- DD: Design Document.
- RASD: Requirements Analysis and Specification Document.
- GM: General Motors is an american multinational corporation that designs, manufactures, markets, and distributes vehicles and vehicle parts.
- API: Application Programming Interfaces
- OnStar: is a subsidiary of General Motors that provide subscription-based communications, in-vehicle security, navigation, and remote diagnostic systems.
- JSF: Java Server Faces, is a Java specification for building component-based user interfaces for web applications.
- MVC: Model View Controller, software design pattern for implementing user interfaces.
- EE: Java Platform, Enterprise Edition (Java EE) is the standard in community-driven enterprise software.
- GPS: Global Positioning System: a global system of U.S. navigational satellites developed to provide precise positional and velocity data and global time synchronization for air, sea, and land travel.

3 Reference Document

- Sample Design Deliverable Discussed on Nov. 2.pdf
- RASD Document
- Assignments AA 2016-2017.pdf

4 Document Structure

- **Introduction:** This section just contains a brief introduction about our Purpose and about the Scope of our Application.
- **Architecture Design:** This section explains our architectural design choices and preferences, it is organized in different chapters:
 - Overview: this section presents the logical and physical layers in our Application
 - High level components and their interaction : In this section we give a global view of the components identified and explains how they interact.
 - Component view : this sections describe each component of the system such database, business layer, presentation layer, client and explain how these component are composed and how communicate between them.
 - Deploying view : this section shows how the components must be developed in according to have the correctly work of the system.
 - Runtime view: in this section we explained using sequence diagram, the main methods used by beans to interact between them.
 - Component interfaces : in this part is described the method and interfaces used to communicate between different component.
 - Selected architectural styles and patterns : there is the description of all the Architectural choices about different system layer.
- **Other design decisions:**
 - Algorithms Design: All the relevant algorithm are described in this section in order to clarify what is the flow of the code. This part is described in a Java like language.
 - User Interface Design: System interface is described by mockup of all devices and the different aspect of mobile and web App.

5 Architectural Design

5.1 Overview

In the architectural design section will be discussed all the system components, both physical and logical level starting from a general view of high level components and their distribution on logical layer (section 2.2). In the next section (section 2.3) we give you more details and more specification about high level system components explaining their main functionality. Section 2.4 is entirely dedicated to explain how these logical level are divided in physical component, so we will explain the best type of physical tiers used in according to obtain robust structure. In section 2.6 we will focus on the interface between different components of the system. The last part (section 2.7) is dedicated to explain and justify the design choices and pattern used in architectural design.

5.2 High Level Components and their Interaction

These are the high level components used in our system:

Database: which is dedicated to storage all Power EnJoy information, so database is the data storage layer, where the system stores permanently all relevant data. Database is one of the most important high level component and for this reason that has to respect the Atomicity, Consistency, Isolation and Durability properties.

Business Layer: this layer is the logic application of the system so is used to execute all process of the system.

Presentation Layer: this is the frontend, able to display and dispose information and data, and also provide possibility to interact with the system, so this layer does not contain any application logic but include web server and reverse proxy.

Mobile application: it refers to presentation layer, and it is used directly by the customer so it provides mobile phone interface (as described in RASD document).

Web application: also web application refers to presentation layer and it provide a web interface (as described in RASD document).

Business Layer: this layer is the logic application of the system so is used to execute all process of the system.

Presentation Layer: this is the frontend, able to display and dispose information and data, and it also provide possibility to interact with the system, so this layer does not contain any application logic but include web server and reverse proxy.

Mobile application: it refers to presentation layer, and it is used directly by the customer so it provides mobile phone interface (as described in RASD document).

Web application: also web application refers to presentation layer and it provide a web interface (as described in RASD document).

As we can see in the graph below the system is composed with four logical layers described above. This design permit to have advantages in terms of performance and security. There is an advantage in performance because each layer has a specific task and functionality to perform, while the advantage in security aspect is due to an assignment of specific security credential to each layer.

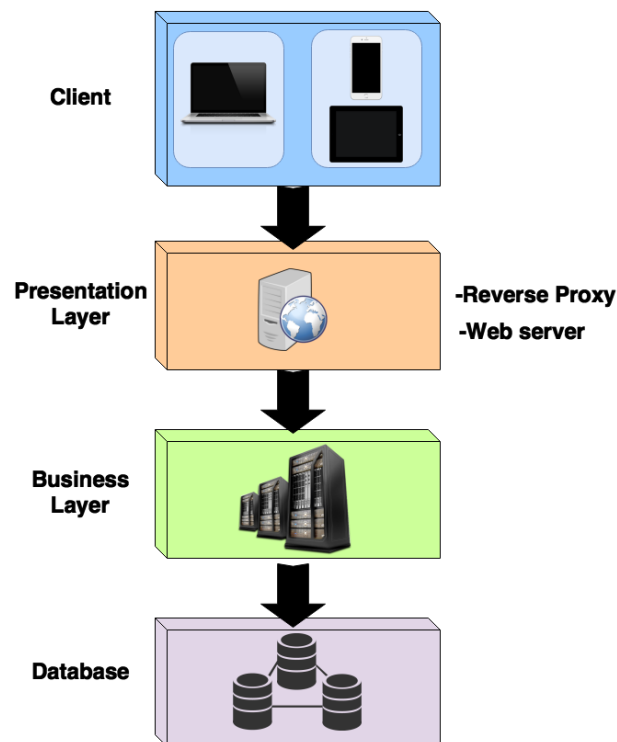


Figure 1: Layer of the system.

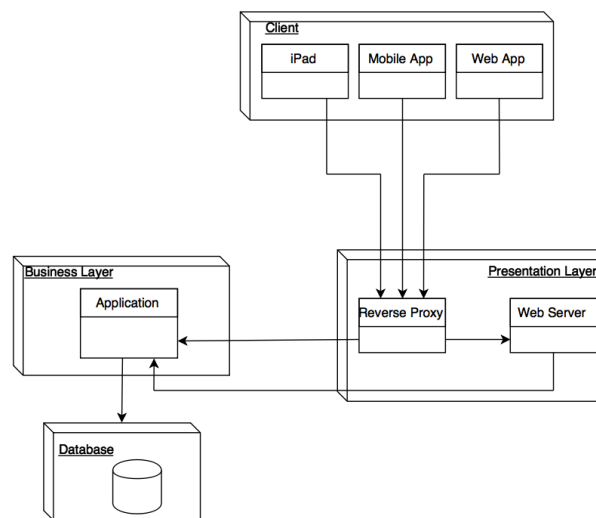


Figure 2: High level components of the system.

5.3 Component view

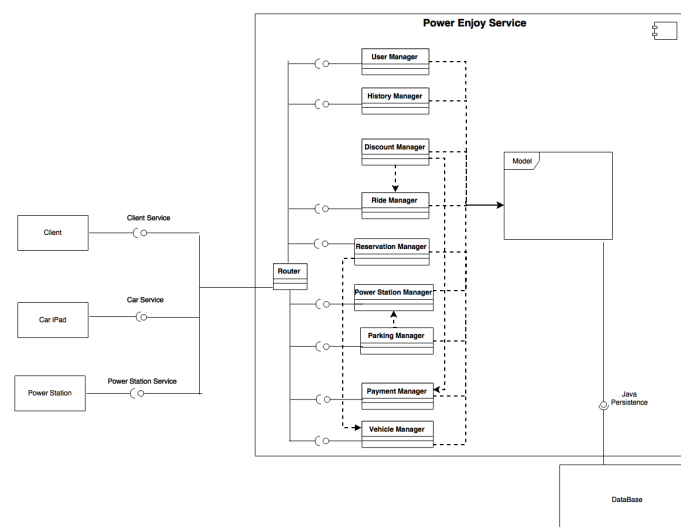


Figure 3: Main Components of the System.

5.3.1 Database

To store all important data of PowerEnJoy System, this Tier will run MySQL in order to support and ensure all important properties such Atomicity, Consistency, Isolation, Durability of database transactions. Database cares to communicate only with logical layer, as described in high level description. Also security aspect will be described about communication used.

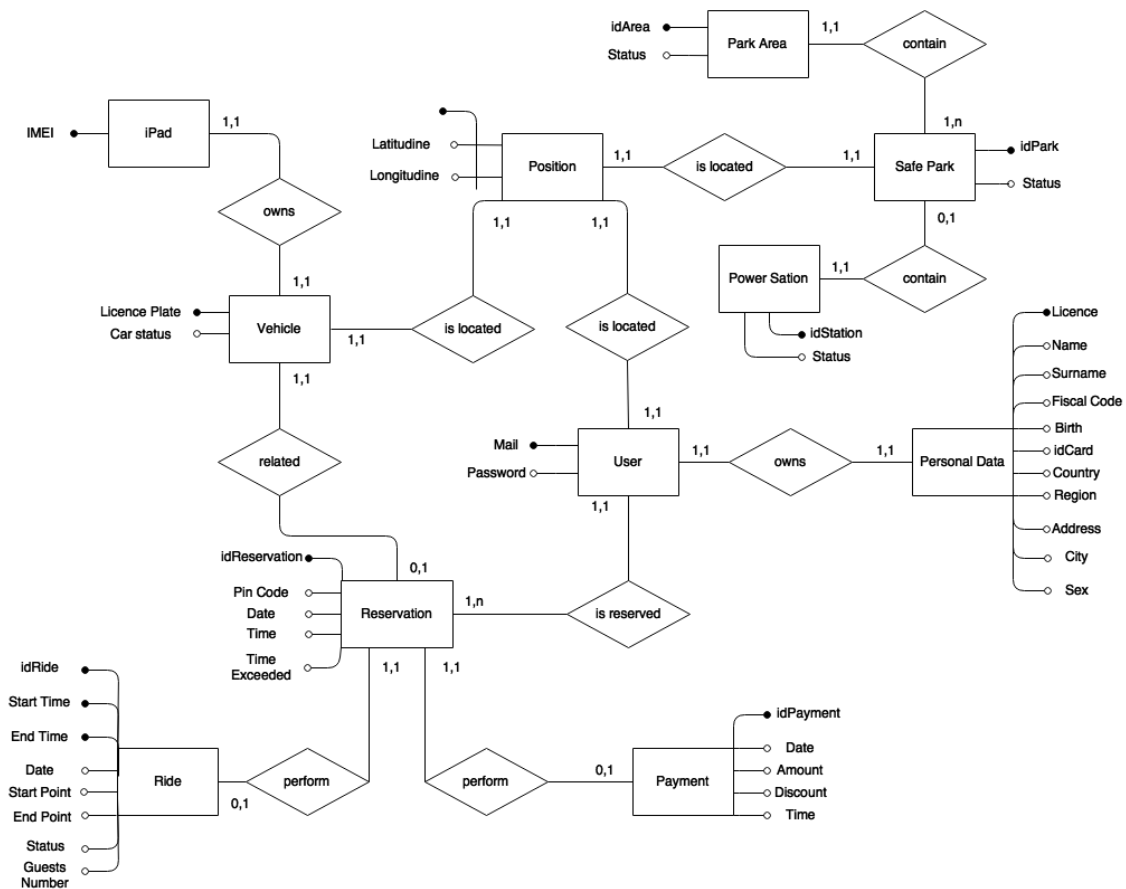


Figure 4: Entity-relationship model representation.

5.3.2 Business Layer

The Business layer is implemented by the Java EE Platform. Business logic will use Java Beans component to encapsulate different object. Logic connects to the database and manage it using Java Persistence application programming interface.

These are the designed beans:

User Manager

This bean allows to manage all the user personal data and also give some additional functionalities such the ability to register into the system, login into the system and verify that user has inserted all the requested data on registering procedure. There will be the possibility also to modify and re-update information.

Ride Manager

This Bean has got an important rule. It controls all the information about a ride, such as start time, stop time, number of guests, distance covered during the trip and the ride status. These check are made in real time, because the user needs to know his ride status showed on iPad. This beans will communicate also with discount manager to apply the discount rules during the payment manager.

Discount Manager

Discount manager bean is used to manage and check, using also ride manager and payment manager, if exists the conditions to apply the appropriate discount. It can be an increment or reduction of the total amount.

History Manager

This beans is used when a customer require to show his history trip. So the history manager elaborate the request and return all the user completed trips. Another functionality of this bean is to maintain history updated.

Reservation Manager

Reservation Manager is a bean that assist user to search and reserve an available car, giving information about it such battery level and distance from user.

Power Station Manager

This bean take care to check and update the power station status, in sense that verify in real-time if a power station is available or not and check if a power station has got an eletrical problems or damage.

Vehicle Manager

Vehicle manager is a Bean that is able to manage all the information about a car, also by invocation of remote command throw GM API. This let the possibility to lock or unlock vehicle doors, start engine ignition, and get information about all the sensors or stats data.

Parking manager

This Bean has to manage the Parking Area, it handles and controls the states of the Parks. Moreover it communicates with the Power Station Manager in order to signal the presence of a Car and with the iPads in the cars to advise the user about parking.

Payment manager

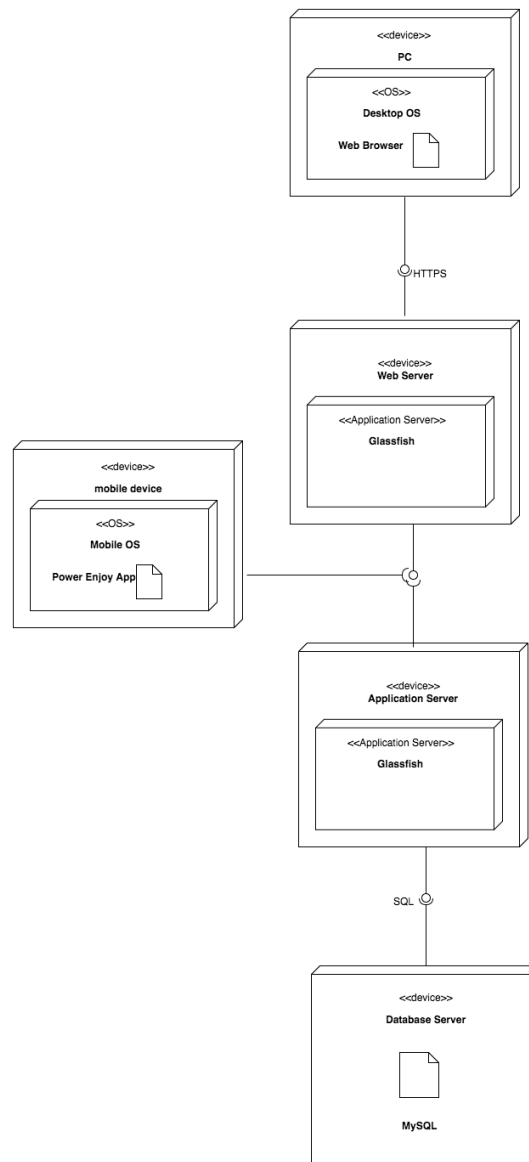
It must handle the payment of the users after every ride and in case of fees

5.3.3 Presentation Layer

The presentation layer is the part used to create and manage the Web App contents. Presentation layer is implemented using Java EE (Enterprise Edition) a platform that provides an application programming interface (API) in sense that, it is a set of subroutine definition, protocols and tools for building application software. The graphical parts of Web App is made using a server-side framework based on MVC (model view controller) called Java Server Faces (JSF). Java Server Faces provide to create a Web interface, in fact the Web App view is made writing an XML file that includes some web programming languages such HTML and/or CSS. The web server inside the presentation layer is run by using GlassFish application server that permit to create portable and scalable enterprise applications. Another component inside the presentation layer is the reverse proxy. Reverse proxy is a logical component that provide to switch the different request that came from Web App and Mobile App. When reverse proxy receive a request from Web App then it is redirect to Web Server, while if the request came from Mobile App it is redirect to Business Layer directly.

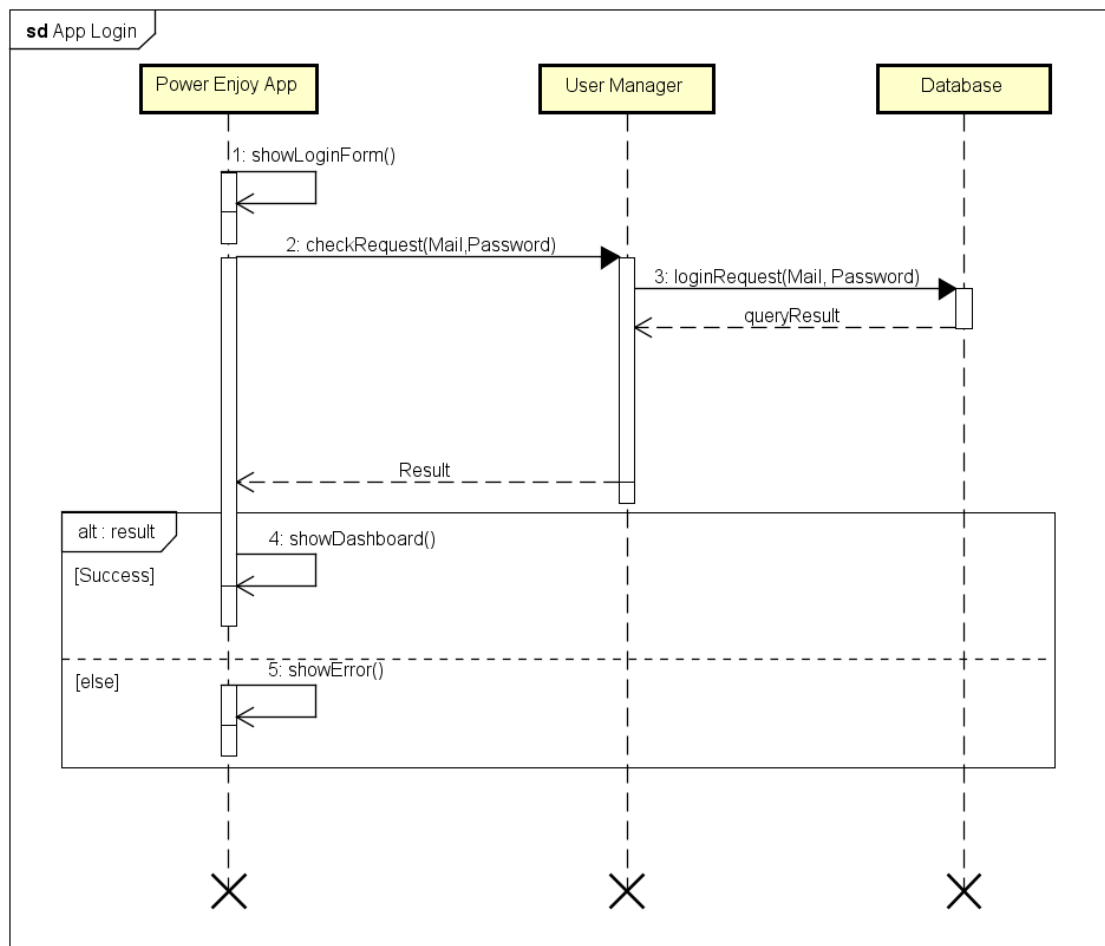
5.4 Deployment view

GlassFish Server is used as the Java EE application server for both the business logic and the web tier.



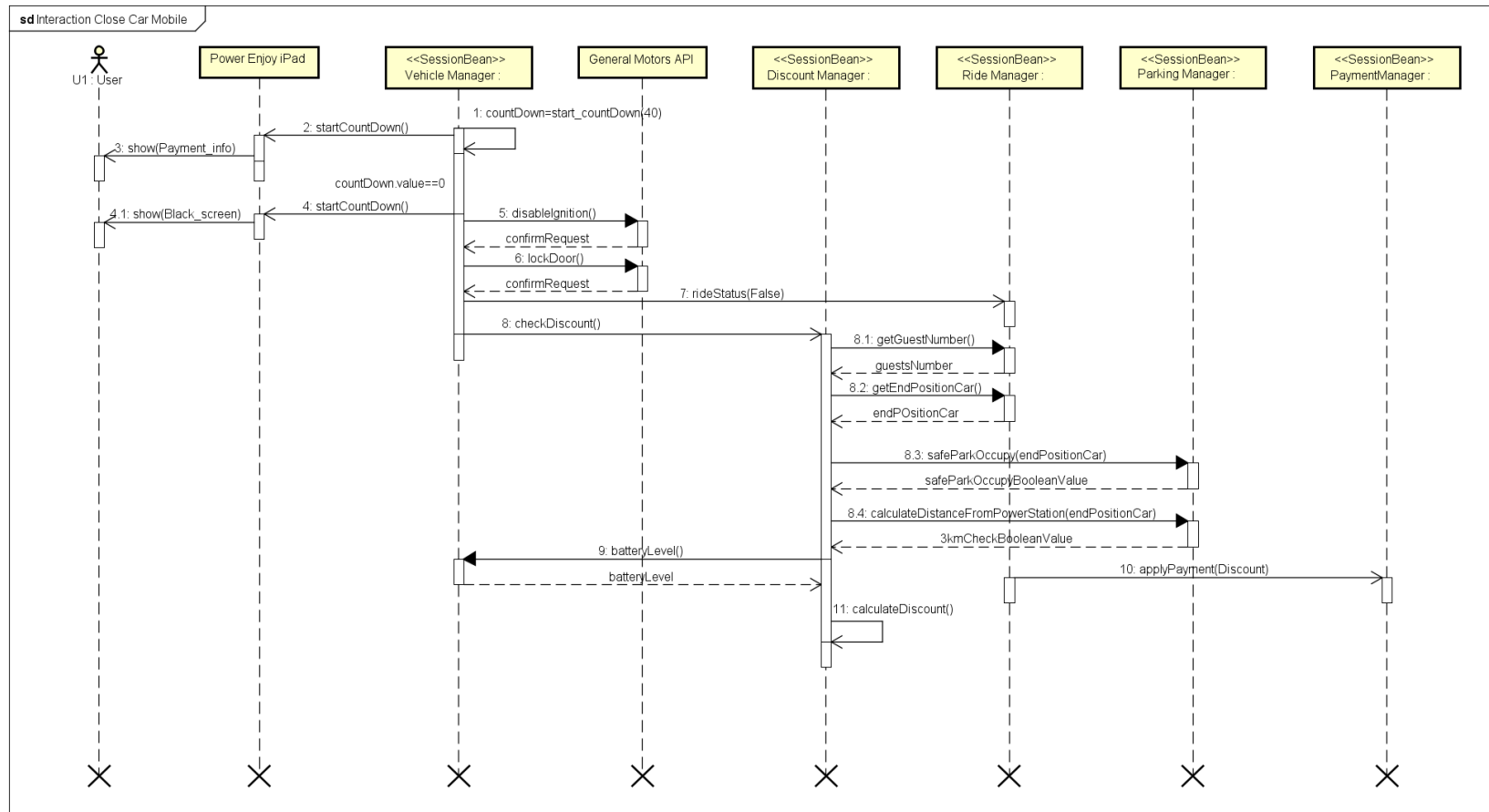
5.5 RunTime View

5.5.1 Login

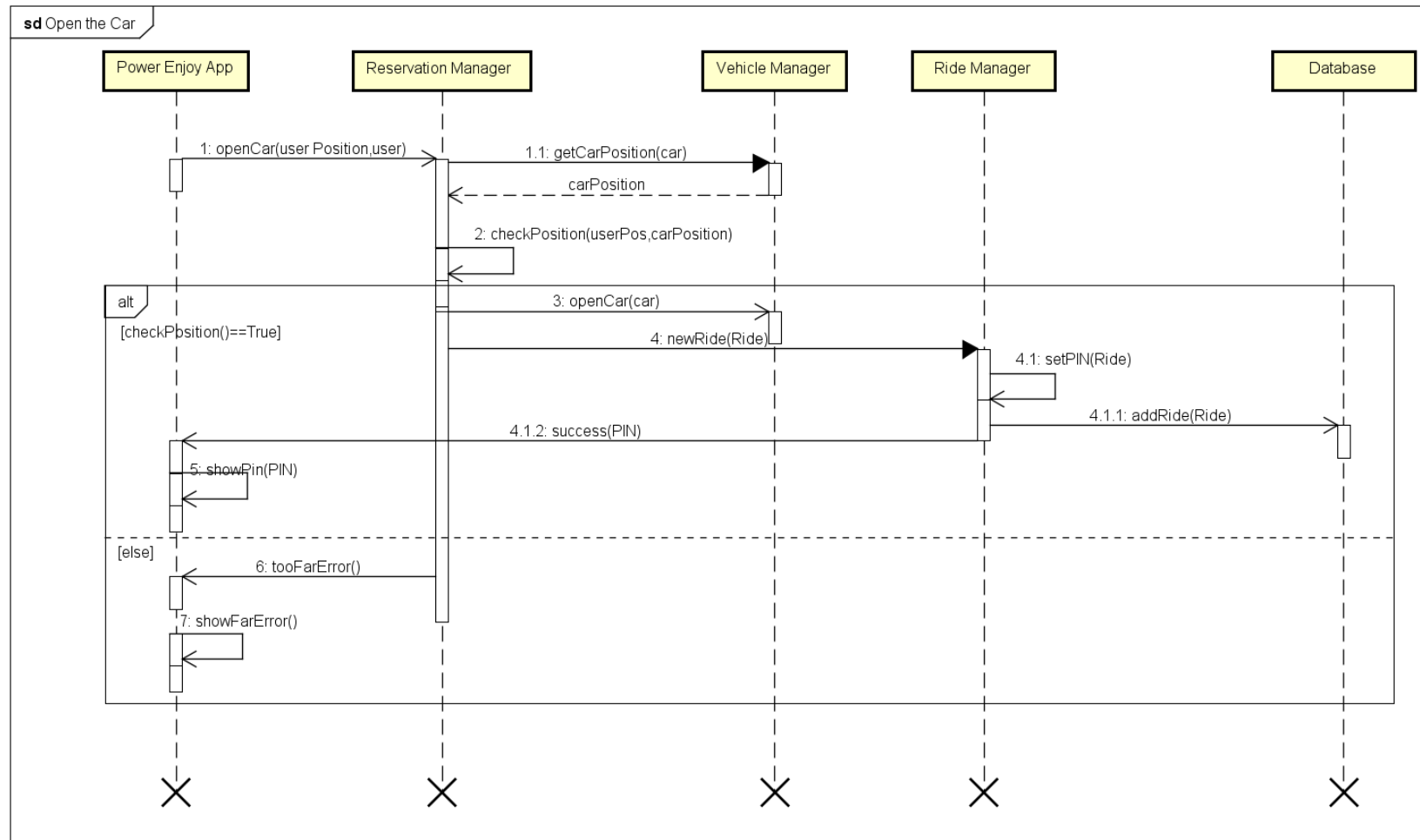


powered by Astah

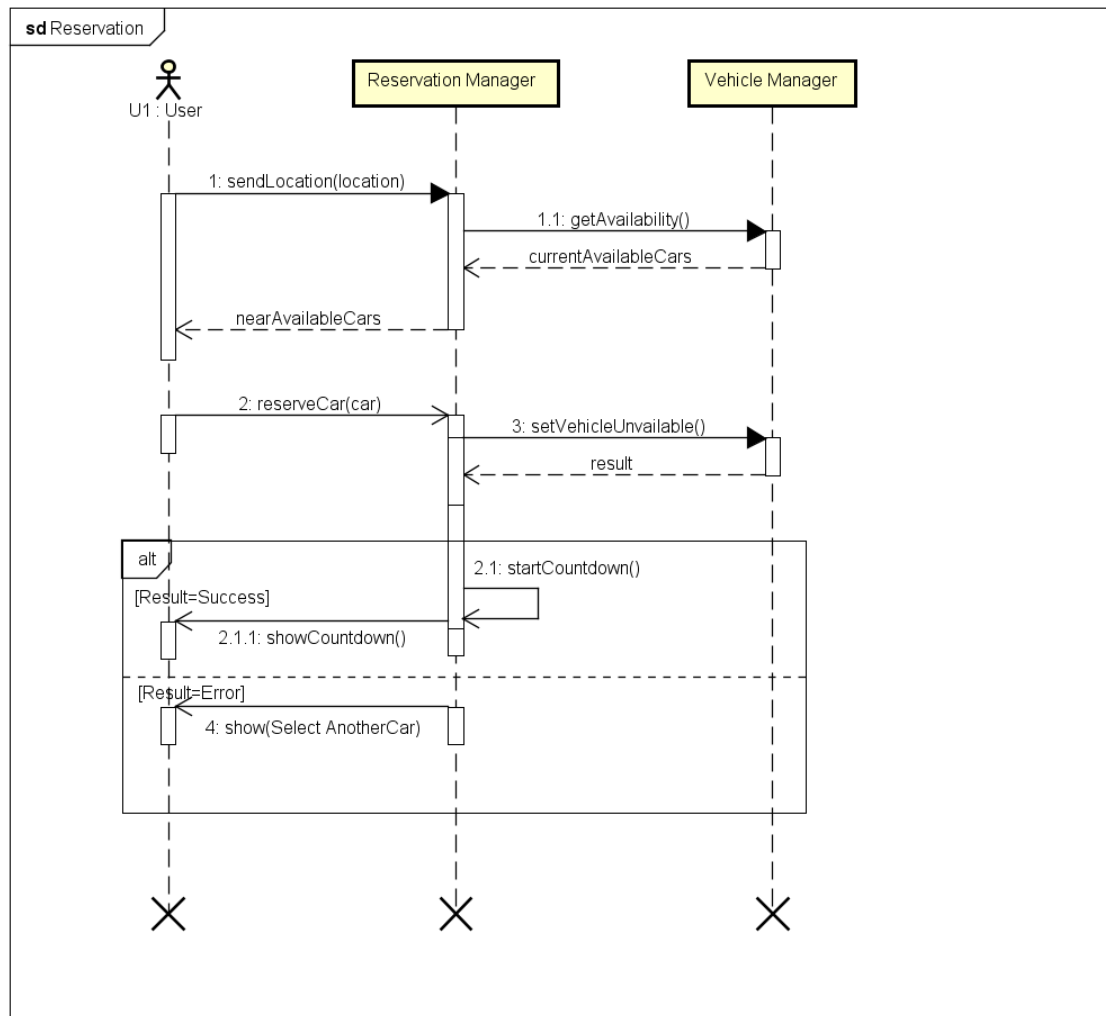
5.5.2 Close car



5.5.3 Open car

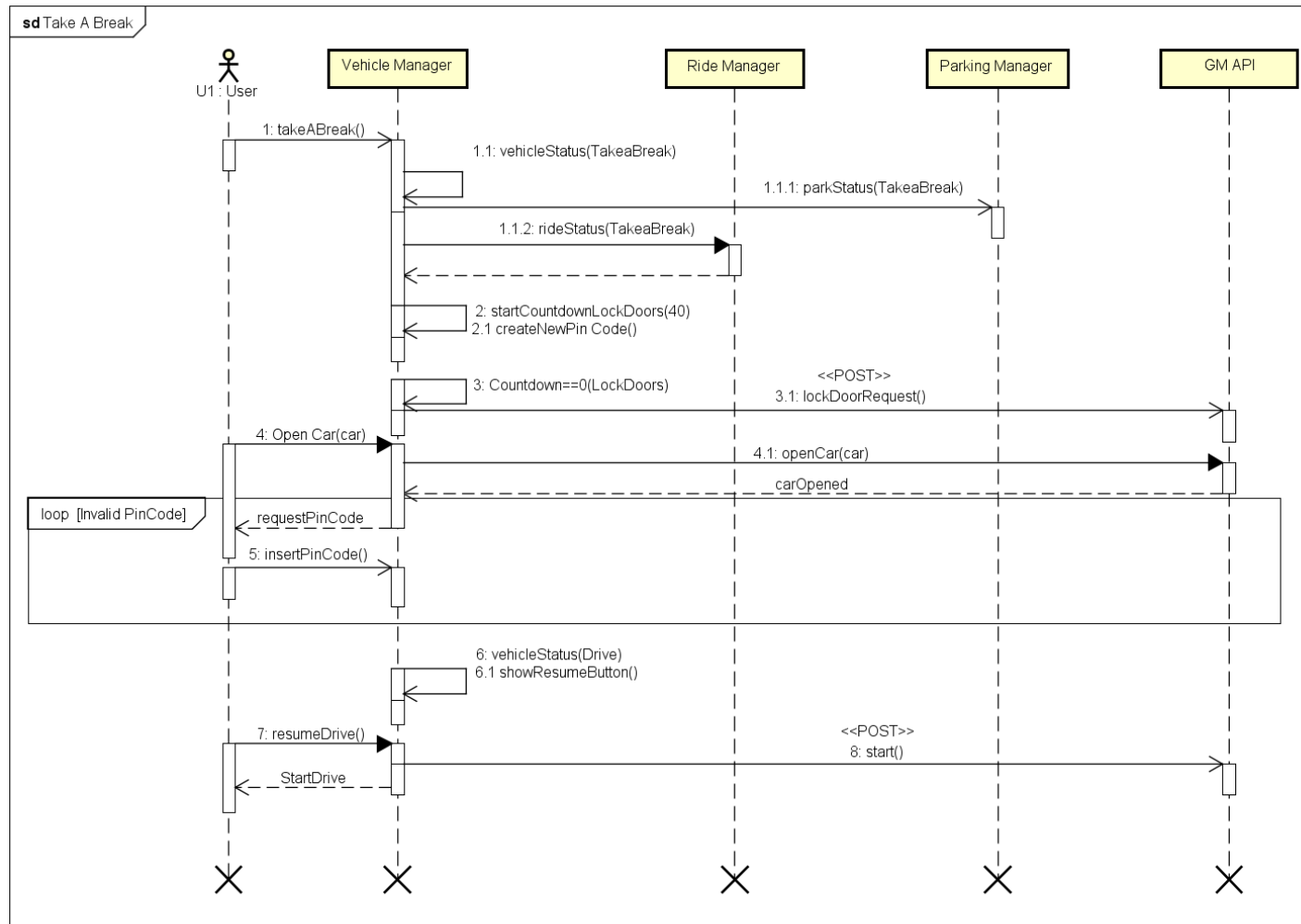


5.5.4 Reservation



powered by Astah

5.5.5 Take a break



6 Component interfaces

In this section we describe the components interfaces and how they communicate between them. Will also include all APIs provide by General Motors Company, used to invoke all operations about car controls.

6.1 UserManager

Functions implemented by **UserManager**:

checkLogin(mail,password)

This function is used from registered customer and permit to log him into the Power EnJoy system. One entered into the system the customer will can use all services offered by Power EnJoy company. This function receive two parameters that are user email and password, and its value of return is a boolean value.

requestNewAccount(data)

This function is used by an unregistered user that want to join into Power EnJoy service. The unregistered user have to fill in the form in WebSite or Mobile App, inserting his personal data necessary for the system. Once the form is entirely compile the unregistered used have to press the submit button to confirm his registration and to send all information to the business logic. This function receive data such input parameter, check if the information are correctly inserted and create a new user entity. It return a boolean value.

modifyAccount(email,informationToModify)

This function is used from a registered customer to modify some personal information such credit card number or dive licence number or else. This function take as input parameters email and the information to modify. Mail address is fundamental to recognise the specific user (key value for database). It return a boolean value.

removeAccount(email)

This function is used by a already registered customer when he decide to stop to used Power EnJoy service removing his account from the company. This function, to identify the specific user that want to eliminate his accounts, take how input parameter the email address (key value for database) and return a boolean value to indicate if the operation was successful completed.

6.2 RideManager

Functions implemented by **RideManager**:

newRide(ride)

This function is used by a logged customer as soon as he open the car with the button shown on his Mobile App. This method permit to create a new ride entity in which will be record all ride information.

Ride Manager also include a large number of “get” functions used by iPad to show all information trip to the customer. These method don’t receive nothing as input parameter, but return only the value described by their name. These function will not explained because their name is already self explanatory.

These functions are:

- `getGuestsNumber()`
- `getEndPoistionCar()`
- `getEndTime()`
- `getTotalDistance()`
- `getStartPosition()`
- `getEndPosition()`

6.3 DiscountManager

Functions implemented by **DiscountManager**:

checkGuestsNumber()

This function is shown by a logged customer at the end of the ride, when he have to do a payment to verify if he is suitable for discount. This function check the presence of almost two others passengers in the car using weight sensors under the seat and verify if the belt are closed. It reads these information directly from GM server, using GM APIs, in particular uses “**diagnostic()**” command.

checkBatteryLevel()

This function is shown in iPad system into a car, to inform the logged customer about battery level. This function interact directly with GM server, using dedicated GM APIs in particular it uses “**diagnostic()**” command.

checkCarPlug()

This function check and shown if the car is connected in a power station, after that the logged customer has clicked on “charge the car” button. This function read this information directly from GM server using given APIs. In particular it uses “**diagnostic()**” command.

checkMoreThan3k(safeParkManager,car)

This function check if the car has been parked in a safe park in which its distance is more than 3 Km from the nearest power station. If this condition is true a charge of 30% on the final price will be applied during the payment session. To calculate the distance between car and power station this method needs car position and safe park position that can be calculated directly from car object and safeParkManager object.

6.4 HistoryManager

Functions implemented by **HistoryManager**:

showHistory(userEmail)

This function is used by logged customer to show his trip history read from Power EnJoy database that contains all rides done by a specific user. This function take an input parameter to recognise a specific user and this function is used both Mobile App and Web App.

6.5 ReservationManager

Functions implemented by **ReservationManager**:

reserveCar(car)

Function used by logged customer when he want to reserve an available car shown on Mobile App map. This function take such a input the car that the customer want to reserve and set the car status on unavailable.

checkPosition(userPos,carPos)

Function used by logged customers that has reserved one car. Once reach the car, this method check if the customer is near enough to the car to open it, because if it not happen the car can't opened. To check the distance between customer and car, this method receive as input user postion and car position.

deleteReservation(car)

Function used by logged customer in two different cases. In the first case there is a direct interaction between Mobile App or Web App and the customer, in sense that if the customer has already reserved a car and he want to delete his reservation, he have to click on button "cancel the reservation". In the second case, the system after one hour from customer reservation, it automatically delete the reservation. This function take such an input parameter the car in which the customer or system wants to remove the reservation.

6.6 Vehicle Manager

The functions described below take such input parameter the car in which apply them, and the functions are:

takeBreak(car)

This function is invoked by a logged customer when he want to use Take A Break service. To invoke this function he needs to click on Take a Break button on the iPad system into the car, after that the system begin the procedure for this functionality as described in the RASD document and sequence diagram.

openCar(car)

This function is invoked by a logged customer when he want to open the car, after reservation procedure, so he needs to click the open the car button only in his mobile phone to start he open the car procedure as described in RASD document and sequence diagram.

closeCar(car)

This function is invoked by a logged customer when he want to close the car at the end of ride. This function starts when the customer click on close car button on iPad system. After clicked on button 40 seconds timer starts and the sequences of process already described on RASD and sequence diagrams is execute.

chargeCar(car)

This function starts when the logged customer that has finished to use the care, to take the discount decide not only to park the car but also plug it into a power station. For doing this functionality the customer once stopped the car in a safe area park with power station have to click the button charge the car button in iPad display.

7 Overall Architecture

We decided to split our System in 3 tiers:

1. Database (MySQL)
2. Application Logic (Business Logic Layer and Presentation Layer)
3. Client (Mobile App, Browser and iPad System)

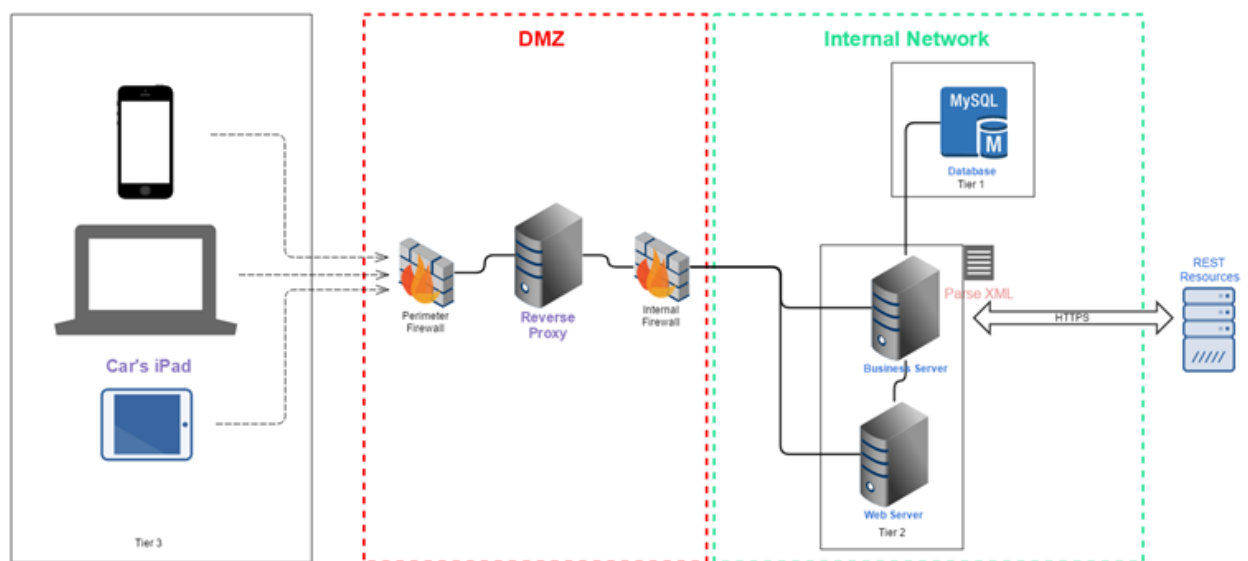


Figure 5: Architectural Overview [4] [5]

Architectural style for the the web app

The web application is designed using HTML and CSS that are included into java faces.

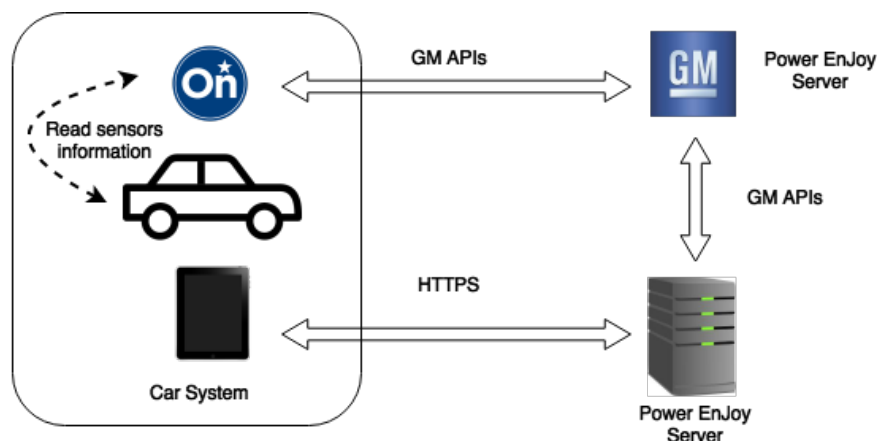
Architectural style for the mobile

The mobile application has been developed for all mobile operative system using a specific SDK: for Apple devices we have used IOS SDK tools, for Android we have used Android SKD tool and for Windows Phone we have used Windows Phone SDK tools.

Architectural style of car system

The car is equipped with an iPad that permit to show to the customer all car's information that came from Power EnJoy server so the iPad app communicate only

with Power EnJoy server using HTTPS protocol. Car is also equipped with OnStar system. OnStar is a physical hardware inside the car that thanks to its internet connection, it can send and receive data that came from GM server using GM APIs. OnStar sends to GM server all information read from sensors put inside the car, such as battery level, weight sensor, seat belt information and more. In conclusion, iPad and OnStar don't communicate directly but the exchange of information happens using Power EnJoy server and GM server.



These systems communicate using different protocols that now we'll present:

Business logic server to database

The Application Server communicates with the database using Java Persistence API, this component abstracts the low level SQL, but operates against entity objects rather than directly with database tables.

Business logic server to App

The communication between the Business logic and the Mobile App and/or Web App is handled through internet communications in particular using HTTPS security protocol to guarantee that reserved information such as password or personal data cannot be read from anyone. In the text below will be describe the most important function implemented by different beans to communicate with the Web App a Mobile App (Front-End).

7.1 Our Server to GM Server(External Server)

Enjoy Application Server communicate to GM Server by using RESTful API: Remote API servers expect to interact with another server, System's architecture must reflect this line. REST architecture style, represent a valid lightweight alternative to RPC and SOAP Web Services, identification is done by OAuth 2.0 protocol. Using this kind of architecture will be helpful some libraries like Jersey, JAX-RS. Application server exchange JSON to interact with GM Servers.

7.2 Design Patterns

Model-View-Controller This separation principle has been widely applied in our application, because it is the most common and the most convenient pattern when it comes to deal with complex application, and allows to design software with the concept of separation in mind. In our application the view developed in two different ways (App/Web App) and both shares the same controller and the same model.

Client-Server

The Client-Server communication model is the most suitable for this application, in fact it allows us to split these two different concepts and to develop a thin client, that must work with low resources, and an heavier part that is handled by the server. Moreover an Application based on the Client-Server is more practical, maintainable and offers advantages in terms of security.

Publish-Subscribe

System will use Publish-Subscribe messaging pattern in order to exchange asynchronous communication to all the different client, included iPad System, and Mobile App. In this way all connected device represent subscribers, and Application Server will be the sender. This compose an important broadcast system in our domain.

7.3 Other Design Decision

Maps Integration

Web and mobile App will use Google Maps API as navigation system and localization service. This choice allow to calculate distance, identify vehicle in the maps. Google Maps API also will help system to organize route, in order to search optimal solution and evaluate a more real ETA, with real time traffic information.

8 Algorithm Design

In this section we'll present the most relevant and critical algorithms in our Application, in particular we'll focus on the advanced features of the System

8.1 Save money option

```
public class saveMoneyOption{
private Position departure;
private Position destination;
private int tollerance;
Area suggestedArea;
Navigator navigator;

private static final int TOLLERANCE=400
//this is the radius of the circle

public saveMoneyOption(Position departure ,
Position destination ,idCar){
this.departure=departure;
this.destination=destination;
navigator=new Navigator(idCar);
}

public Area findArea(){
return Area.getMoreFreeArea(destination ,TOLLERANCE);
}
public startNavigator(){
navigator.start(this.findArea().getPosition());
}
```

description of getArea method:

```
Area getMoreFreeArea(Position destination , int tollerance){
int min=10000; // high number
Area areaChoice= new Area();
for(int i=0;i<area.size();i++){
if(area.get(i).powerStationOccupy()<min)
min=area.get(i).powerStationOccupyInArea()
areaChoice=area.get(i);
}
return areaChoice;
}
```

8.2 Discount algorithm

```
public class DiscountManager{
    Vehicle vehicle;
    SafePark safePark;
    Ride ride;

    public DiscountManager(Vehicle vehicle , SafePark safePark , Ride ride){
        this , vehicle=vehicle;
        this.ride=ride;
        this.safePark=safePark;
    }

    float discountPercentage(){
        float discount=0;

        if (moreThan3km() || checkLowBatteryLevel())
            discount=0.3

        if (checkCarInCharge())
            discount -=0.3;
        else
            if (checkBatteryLevel())
                discount -=0.2;
            else
                if (checkGuestDiscount())
                    discount -=0.1;

        return discount;
    }
}
```

8.3 Open the car algorithm

```
public class ReservationManager{
private ArrayList<Reservation> reservations;
private VehicleManager vehicleMg;
private

public ReservationManager(VehicleManager vehicleMg){
this.vehicleMg=vehicleMg;
}

public addRes(Reservation res){
reservations.add(res);
}

public boolean openCar(Position userPosition, User user){
boolean result;
for(i=0;i<reservations.size();i++){
if(reservations.get(i).getUser().equal(user)){
if(checkPosition(vehicleMg.getCarPos(userPosition,
reservations.get(i).getCar()), userPosition){
vehicleMg.openCar(reservations.get(i).getCar());
reservations.get(i).newRide();
result=true;
}else{
result=false
i=reservations.size();
}
}
}
return result
}

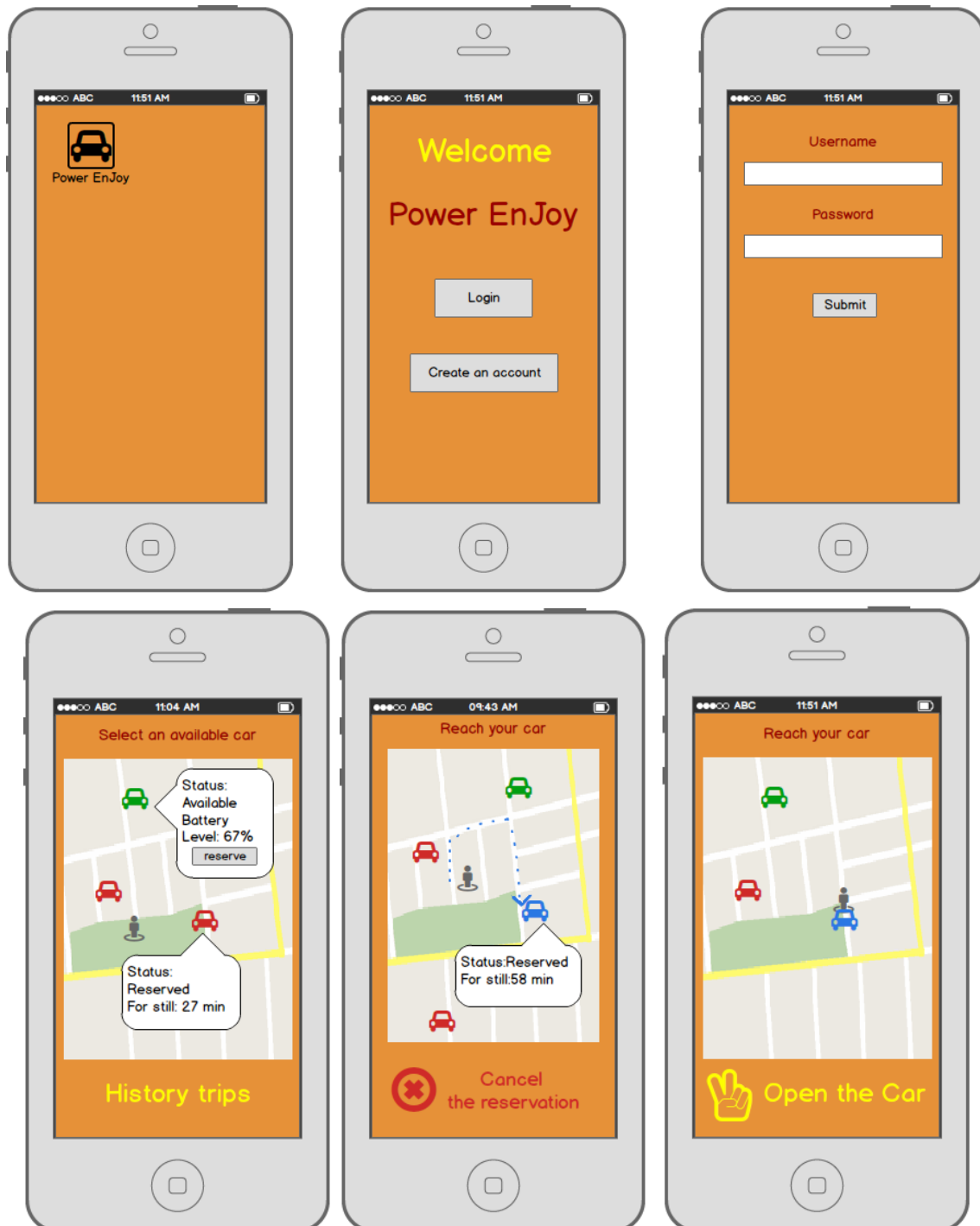
private boolean checkPostion(Position p1, Position p2){
if(p1!=p2)
return false;
else
return true;
}
```


9 User interface design

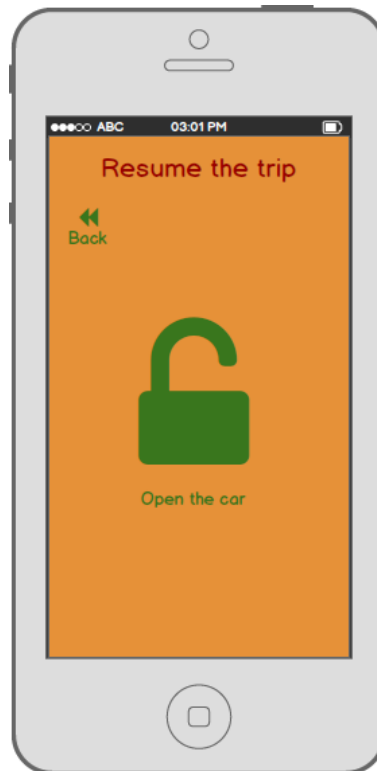
9.1 iPad System



9.2 Mobile App







10 Web App

Power EnJoy

An electric Car Sharing System

[Home](#) |
 [About](#) |
 [FAQ](#) |
 [Advertise](#) |
 [Contact](#)

[Home](#)
[Map](#)
[Vehicle](#)
[Payments](#)
[Contact](#)
[Create an Account](#)

Login:

Best Way to Move around the City

Save Money

11 November 2008

Using our system you will save money, it's easy to start. Register and start Driving!

Guidelines - Respect Environment

No Pollution!


Social

- Add Us On Facebook
- Add Us On Twitter
- Add Us On Myspace

Copyright Your Company © 2016 All Rights Reserved.

Power EnJoy

An electric Car Sharing System



[Home](#) | [About](#) | [FAQ](#) | [Advertise](#) | [Contact](#)

Home

Map

Vehicle

Payments


Account

Welcome, username


Logout

Look for a Car

Address:




Cars Nearby:




Distance: 275 meters

Battery: 20%




Reserved



Distance: 150 meters

Battery: 80%




Reserve it

Copyright Your Company © 2016 All Rights Reserved.

Home | About | FAQ | Advortise | Contact

Power eJoy

An electric Car Sharing System



Registration

Fill all required data:

First Name:

Last Name:

Email:

Repeat Email:

Password:

Repeat Password:

Sex: ☐ Male ☐ Female

Birth:

City:

Region:

ZIP Code:

Tax Code:

Identity Card:

N° Identity Card:

Release Date:

Expiration Date:

Cancel

Register

Copyright Your Company © 2016 All Rights Reserved.

[illegible]

11 Requirements Traceability

In this part will be discuss and represent how we mapped the design document (DD) element with requirements analysis specification document (RASD) both functional and non functional requirements.

11.1 Functional requirements and components

The table below show how we have combined the component of DD with functional requirements defined in RASD.

- [G1] System checks the user's identity through the login procedure.
 - User Manager
- [G2] System allows logged customers to see available vehicles and Trip History.
 - User Manager
 - Reservation Manager
 - History Manager
- [G3] Customers will be able to reserve chosen car for 1 hour before they pick it up.
 - Reservation Manager
 - Vehicle Manager
- [G4] User that exceed the pickup time will be charged 1 euro of fee and the car come back available.
 - Reservation Manager
 - Payment Manager
- [G5] User that reaches the reserved car is able to open it using the app.
 - Ride Manager
 - Vehicle Manager
 - Reservation Manager
- [G6] The system during the trip shows details about the current charge, the safe area for parking and for recharging the vehicles.
 - Ride Manager
 - Parking Manager
 - Power Station Manager

- [G7] The system computes automatically charges and discounts related to the user behaviour.
 - Discount Manager
 - Payment Manager
- [G8] System show on the display an ETA.
 - Ride Manager
- [G9] System will include assistance service h24.
 - Call Center Service

12 RASD modification

- Modify Goal [G2].
- Modify UML Diagram.
- Correct some grammatical errors.

13 Used tools

The tools used to create this RASD document are:

- Google Drive: used to write simultaneously text parts.
- GitHub: to deliver the document.
- Eclips: to write java code.
- Draw.io (Google extension): for UML schema and User Case schema.
- Gliffy: to design diagrams.
- TeXstudio: LaTeX editor.
- Skype: used to do video-conference .

14 Hours of work

We have often worked together at university or using skype during the week-end.

Gabriele Bressan

- 25/11/16 3h
- 26/11/16 3h
- 27/11/16 3h
- 1/12/16 4h
- 2/12/16 5h
- 3/12/16 4h
- 4/12/16 2h
- 7/12/16 3h
- 8/12/16 2h
- 9/12/16 4h
- 11/12/16 2h

Simone de Santis

- 25/11/16 4h
- 27/11/16 2h
- 29/12/16 4h
- 2/12/16 3h
- 3/12/16 3h
- 4/12/16 4h
- 5/12/16 5h
- 8/12/16 3h
- 9/12/16 4h
- 10/12/16 4h

Pietro Di Marco

- 26/11/16 4h
- 27/11/16 3h
- 28/11/16 4h
- 29/11/16 3h
- 2/12/16 3h
- 3/12/16 4h
- 4/12/16 4h
- 6/12/16 4h
- 8/12/16 3h
- 9/12/16 4h
- 11/12/16 3h

References

- [1] Security Aspect: ref: RFC6749
- [2] REST Arch (University of California): UCI
- [3] GM Developer Site(only for registered developer): GM
- [4] Example of Reverse Proxy and possible configuration with Load Balancer: McAfee Web Gateway
- [5] Reverse Proxy Servers Overview in a real domain: Computer Associates International Inc. Documentation