Politecnico di Milano, A.A. 2016/2017

Software Engineering 2: PowerEnJoy
Design **D**ocument

Binosi Lorenzo 876022

December 7, 2016

Contents

1 Introduction

1.1 Purpose

The Design Document has the purpose to provide a functional description of the system. It defines the design of the software architecture, its components and their interactions, provided with the used algorithms and the user interfaces design.

The document is written for project managers, developers, testers and quality assurance. It can be used for a structural overview to help maintenance and further development.

1.2 Scope

PowerEnJoy is a large-scale car-sharing service. It requires to perform excellently, to be secure, easily modifiable and as available as possible. Minding these necessities, in the following chapters will be shown how the system is structured and and which were the reasons that led to such decisions.

1.3 Definitions, Acronyms, Abbreviations

DD: Design Document.

RASD: Requirements Analysis and Specification Document.

System: the whole software system to be developed, comprehensive of all its parts.

REST: REpresentational State Transfer. It's an architectural style and an approach to stateless communications, used in the development of client-server systems.

GUI: Graphical User Interface.

ACID: Atomicity, Consistency, Isolation, Durability. A set of properties of database transactions.

1.4 Reference Documents

This document refers to the project rules of the Software Engineering 2 project [1] and to the DD assignment [2].

1.5 Document Structure

This document is structured in five parts:

section 1: Introduction. This section provides general information about the DD document and the used terms.

section 2: Architectural Design. This section shows the software architecture of the system, with its components and their interactions.

section 4: Algorithm Design. This section will present and discuss in detail the algorithms designed for the system functionalities, independently from their concrete implementation.

section 3: User Interface Design. This section shows how the user interface will look like and behave, by means of concept graphics and UX modeling.

section 5: Requirements Traceability. shows how the requirements in the RASD [3] are satisfied by the design choices of the DD.

2 Architectural Design

2.1 Overview

The PowerEnJoy's software architecture will be described starting from an high level view up to an in detail view of a specific system component. Going into details will be discussed the roles of these components and how can they interact through several interfaces.

A static and dynamic viewpoint are provided in order to better explaining how the architecture works through its components.

2.2 High Level View

In order to explain the role of the system's components, it's better considering the layers involved for each component. There are 3 different distinguishable layers:

Presentation: this layer provides a data processing for a GUI and/or a GUI itself.

Application: this layer provides one or more logic functionalities. Functionalities could be core, if they are performing an essential function for the system, or peripheral, if they are managing interactions between core functionalities or interfaces between system's components.

Data: this layer provides a way to store and retrieve the essential data for the system.

The main components of the system are the following:

Database: a data layer that supports a RDBMS.

Application server: an application layer responsible for the whole application logic of the system. All the policies, the algorithms and the computation are performed here. This layer offers a service-oriented interface.

Web server: a presentation layer responsible for translating HTTPS requests into API requests and API responses into HTTPS responses.

Mobile application: a presentation layer directly connected to the application server.

Car applications: Cambiarea presentation and application layer that provide utilities to the user and an interface to the server in order to manage the car.

User's browser: a presentation layer consisting in a web browser. Not under control of the system.

The following figure shows all the components of the system, highlighting the logic layers for each component, grouped by the physical tiers.

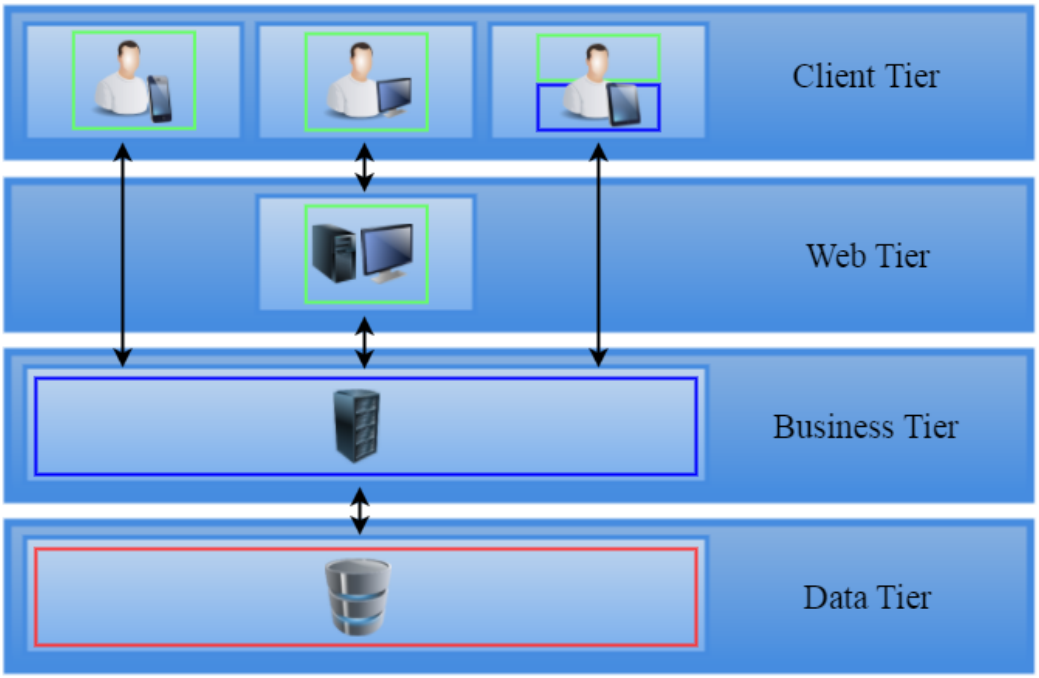

Figure 2.1: High level view of the PowerEnJoy architecture.

2.3 Component View

2.3.1 Database

The server database of the system runs MySQL Community Edition 5.7 with InnoDB as default storage engine. InnoDB provides the standard ACID-compliant transaction features.

The database is only and directly connected to the application server through the standard network interface, described in **??**.

The following properties are satisfied:

- passwords are not saved in plain-text, but they are salted with 8 random bytes, added to the password, and encrypted using the SHA-1 algorithm. The 8 random bytes are different for each password and obviously stored.
- access to the data must be granted only to authorized users possessing the right credentials.
- every software component that needs to access the DBMS must do so with the minimum level of privilege needed to perform the operations.

All the persistent application data is stored in the database. The relational model is illustrated in the following figure.

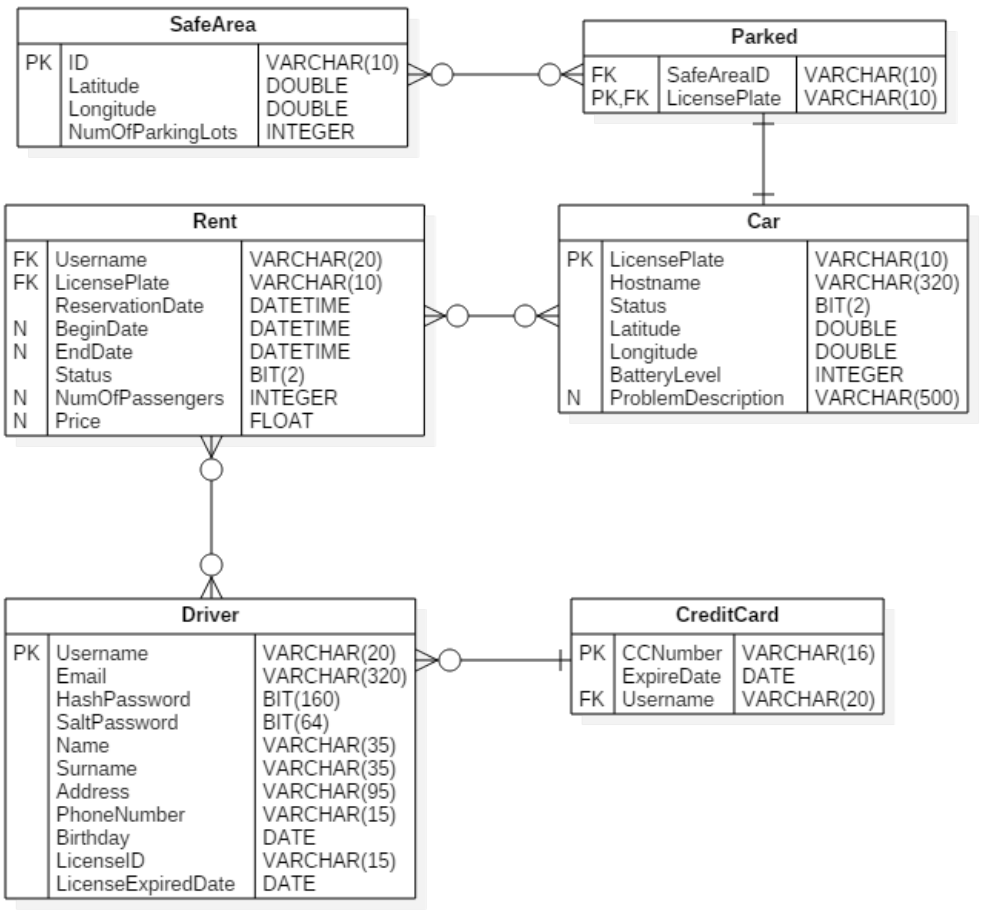

Figure 2.2: Relational model of the database schema.

Several triggers are implemented to manage some exceptions. For instance:

- the registration is cancelled if a driver has never logged into the system for 10 days since his/her registration.
- the rent is aborted if a driver doesn't unlock the reserved car within 1 hour. Price of the rent is setted to 1 euro.
- a report of assistance to a car is generated if it have been passed more than 10 minutes from the last car heartbeat.

2.3.2 Application Server

The application servers are implemented using Java EE; they runs on GlassFish server.

The business logic is implemented by custom-built **Enterprise Java Beans (EJB)**. Cosa sono e come sono formati. Given that the state of users, cars and rents is stored in the DB, all the EJB are stateless.

Parlare delle servlet

The access to the DBMS is not implemented with direct SQL queries: instead, it is completely wrapped by the **Java Persistence API (JPA)**. The object-relation mapping is done by entity beans. They are simple java classes without a constructor and fulfilled by getter and setter methods for each attributes, that correspond to fields of a database table. EJBs use those entity beans in order to read and write on the database.

FARE CLASSI UML

Session Beans used for the application server are shown in Figure 2.3.

2.3.2.1 UserManager

This bean manages all the user management features: user registration, user deletion, user profile editing. User registration provides a function that sends an email, with a password and some guidelines, after the registration.

2.3.2.2 HistoryManager

This bean allows users to fetch the history of their past and current rents.

2.3.2.3 ReservationManager

This bean allows users to get information about the available cars.

2.3.2.4 DriverRentManager

This bean allows a user to create a new rent, unlock a reserved car and fetch information about the current rent.

2.3.2.5 CarHeartBeat

This bean manages information coming from cars. It also provides a function responsible for closing cars.

2.3.2.6 CarRentManager

This bean manages rental features: current rental charge, nearby safe areas.

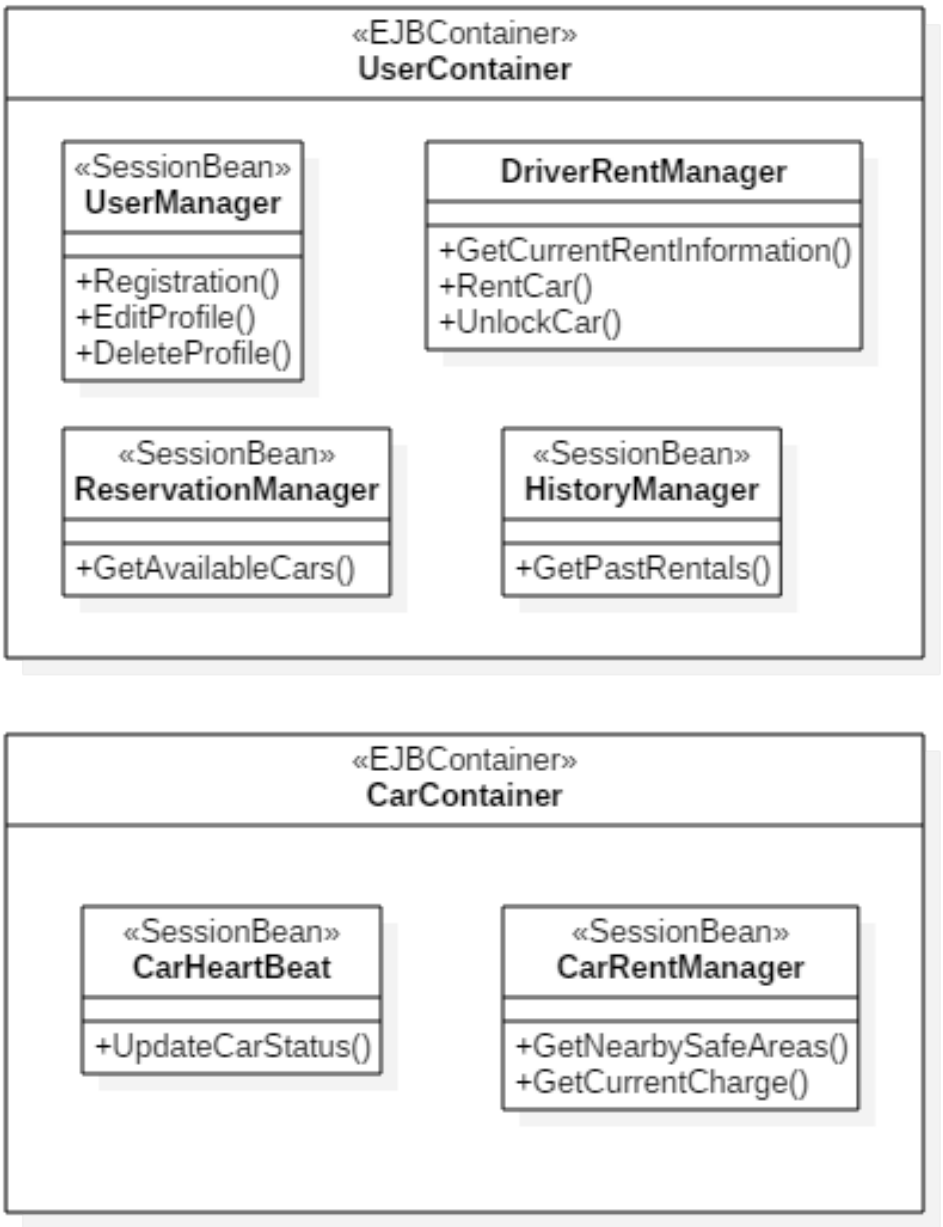

Figure 2.3: Session EJBs implemented server side.

2.3.3 Web Server

The web server runs on GlassFish server. It's implemented using **JSP**; java EE web components useful when the goal of the web server is to translate RESTful API requests/responses into HTTP/HTTPS requests/responses. Thanks to JSPs, developers can focusing better on the GUI.

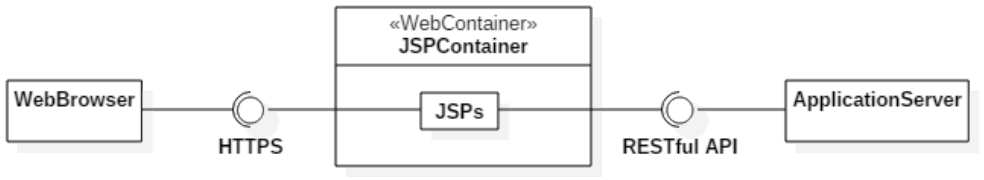

Figure 2.4: The components and the interfaces of the web tier.

Even if REST is a stateless architectural style, JSPs lead users to a statefull communication saving the session; in order to avoid further insertion of user's credentials.

2.3.4 Car Applications

Every car hosts a computer with a linux distribution of Ubuntu 16.10 as operating system. In any OS is installed openssh-server, in order to access to the car remotly, and two applications:

HeartBeat: this application is always running and sends, every 5 seconds, information about the current state of the car to the application server through a single RESTful API call.

CarGUI: this application is started everytime the car get unlocked, and closed everytime the car get locked. It asks, through API calls, for user information, user's current charges and nearby safe areas, in order to display those information on the GUI.

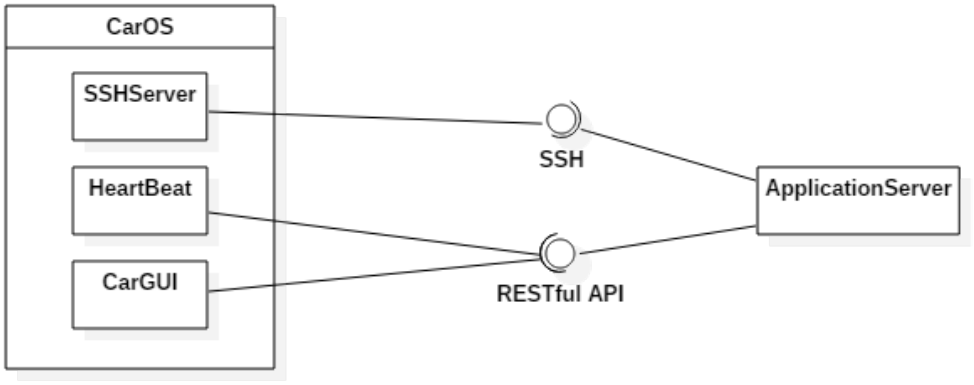

Figure 2.5: The components and the interfaces of the car client tier.

2.3.5 Mobile Application

The mobile client implementation depends on the specific platform. The iOS application is implemented in **Swift** and mainly uses **UIKit** framework to manage the UI interface. Instead, the Android application is implemented in **Java** and mainly uses **android.view** package for graphical management.

The application core is composed by a controller which translates the inputs from the UI into remote functions calls via RESTful APIs. The controller also manages the interaction with the GPS component using **CoreLocation** framework in iOS app and **LocationListener** interface in the Android one.

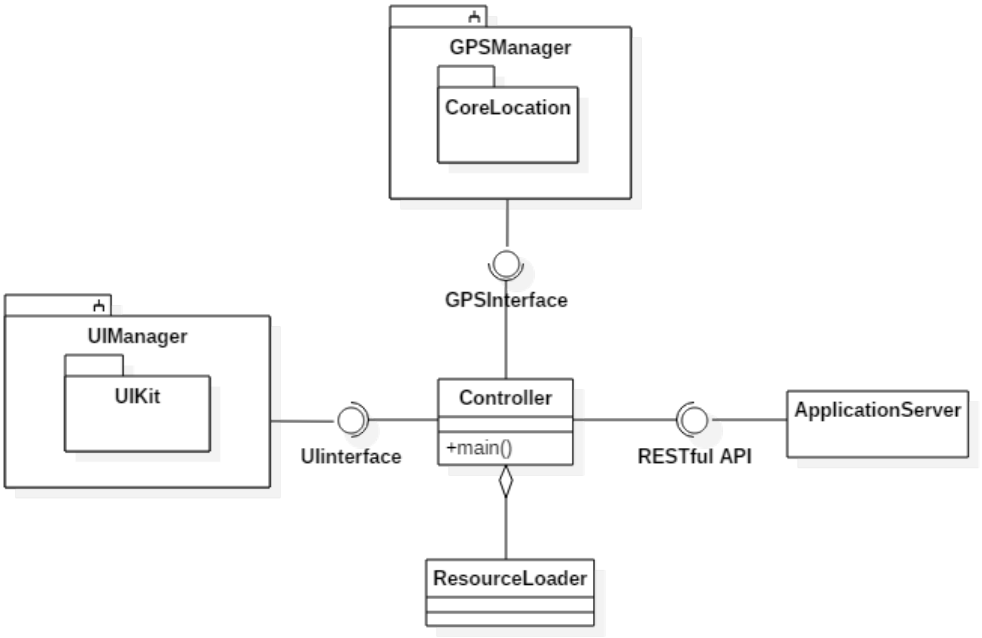

Figure 2.6: The components of the iOS application.

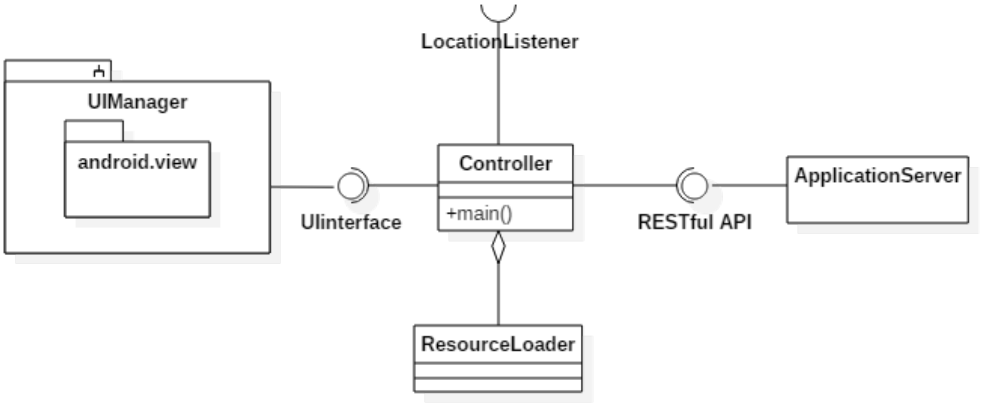

Figure 2.7: The components of the Andorid application.

2.4 Component Interfaces

2.4.1 Front-ends to Application Server (API)

As already said, the front-ends applications comunicate with the application server through **RESTful API** over the **HTTPS** protocol. They are provided by the application server and use **JSON** as data representation language.

The detailed RESTful API is described in the following pages.

2.4.1.1 Creates user

POST /api/user/add-user — Creates user (REST)

Request Classes

Request Value: User

User – This object contains all the user's information.

Name	Data Type	Description
username	string	The username of the user.
password	string	The password of the user.
name	string	The name of the user.
surname	string	The surname of the user.
address	string	The address of the user.
phoneNumber	string	The mobile phone number of the user.
birthday	string	The birthday of the user.
creditCard	CreditCard	The credit card associated to the account.
drivingLicense	DrivingLicense	The driving license of the user.

CreditCard – This object contains credit card information.

Name	Data Type	Description
creditCardNumber	string	The number of the credit card.
expireDate	string	The credit card date of expiry.

DrivingLicense – This object contains driving license information.

Name	Data Type	Description
drivingLicenseID	string	The ID of the driving license.
expireDate	string	The driving license date of expiry.

Response Errors

HTTP Status Code	Reason
400	Bad Request
403	Forbidden – SSL or TLS required
422	Unprocessable Entity – Error reasons
500	Internal Server Error
503	Service Unavailable

2.4.1.2 Retrieves user's salt bytes

GET /api/user/{username}/salt-bytes — Retrieves user's salt bytes (REST)

Response Classes

Return Value: SaltBytes

SaltBytes – This object contains the user's salt bytes.

Name	Data Type	Description
saltValues	List[byte]	Salt bytes to use in the hash function.

Response Errors

HTTP Status Code	Reason
400	Bad Request
403	Forbidden – SSL or TLS required
404	Not Found – User not found
500	Internal Server Error
503	Service Unavailable

Path Parameters

Name	Data Type	Description
username	string	The username of the user.

2.4.1.3 Deletes user

DELETE /api/user/delete-user — Deletes user from the system (REST)

Response Errors

HTTP Status Code	Reason
400	Bad Request
403	Forbidden – Authentication error
403	Forbidden – SSL or TLS required
500	Internal Server Error
503	Service Unavailable

Query Parameters

Parameter Name	Value Data Type	Value Data Description
username	string	The username of the user.
hashed_password	string	The hash of the user's salted password in hexadecimal notation.

2.4.1.4 Retrieves user's rental logbook

GET /api/logbook/past-rents — Retrieves user's rental logbook (REST)

Response Classes

Return Value: Logbook

Logbook – This object contains the user's rental logbook.

Name	Data Type	Description
rents	List[Rent]	This list of the past rents.

Rent – This object contains the information about a single rent.

Name	Data Type	Description
licensePlate	string	The license plate of the rented car.
beginDate	string	The rental start date.
endDate	string	The rental end date.
cost	float	The rental cost.
isPayed	boolean	Is payed or not.

Response Errors

HTTP Status Code	Reason
400	Bad Request
403	Forbidden – Authentication error
403	Forbidden – SSL or TLS required
500	Internal Server Error
503	Service Unavailable

Query Parameters

Parameter Name	Value Data Type	Value Data Description
username	string	The username of the user.
hashed_password	string	The hash of the user's salted password in hexadecimal notation.

2.4.1.5 Retrieves available cars

GET /api/reservation/search-cars/{latitude}/{longitude} Retrieves available cars (REST)

Response Classes

Return Value: AvailableCars

AvailableCars – This object contains information about the available cars.

Name	Data Type	Description
cars	List[Car]	The list of the available cars.

Car – This object contains information about a car.

Name	Data Type	Description
licensePlate	string	The license plate of the car.
batteryLevel	string	The battery level of the car.
position	Position	The position of the car.

Position – This object contains geographic latitude and longitude.

Name	Data Type	Description
latitude	double	The latitude value.
longitude	double	The longitude value.

Response Errors

HTTP Status Code	Reason
400	Bad Request
403	Forbidden – Authentication error
403	Forbidden – SSL or TLS required
500	Internal Server Error
503	Service Unavailable

Path Parameters

Name	Data Type	Description
latitude	double	The latitude value of the user's position.
longitude	double	The longitude value of the user's position.

Query Parameters

Parameter Name	Value Data Type	Value Data Description
username	string	The username of the user.
hashed_password	string	The hash of the user's salted password in hexadecimal notation.

2.4.1.6 Reserves car

GET /api/reservation/{licensePlate}/reserve — Reserves car (REST)

Response Errors

HTTP Status Code	Reason
400	Bad Request
403	Forbidden – Authentication error
403	Forbidden – SSL or TLS required
404	Not Found – Car not found
422	Unprocessable Entity – Error reasons
500	Internal Server Error
503	Service Unavailable

Path Parameters

Name	Data Type	Description
licensePlate	string	The license plate of the car that the user wants reserve.

Query Parameters

Parameter Name	Value Data Type	Value Data Description
username	string	The username of the user.
hashed_password	string	The hash of the user's salted password in hexadecimal notation.

2.4.1.7 Retrieves current reservation

GET /api/rent-operations/current-reservation — Retrieves current reservation (REST)

Response Classes

Return Value: CurrentReservation

CurrentReservation – This object contains information about the current reservation.

Name	Data Type	Description
car	Car	The reserved car.
reservationDate	string	The rental reservation date.

Car – This object contains information about the reserved car.

Name	Data Type	Description
licensePlate	string	The license plate of the car.
batteryLevel	string	The battery level of the car.
position	Position	The position of the car.

Position – This object contains geographic latitude and longitude.

Name	Data Type	Description
latitude	double	The latitude value.
longitude	double	The longitude value.

Response Errors

HTTP Status Code	Reason
400	Bad Request
403	Forbidden – Authentication error
403	Forbidden – SSL or TLS required
404	Not Found – Reservation not found
500	Internal Server Error
503	Service Unavailable

Query Parameters

Parameter Name	Value Data Type	Value Data Description
username	string	The username of the user.
hashed_password	string	The hash of the user's salted password in hexadecimal notation.

3 User Interface Design

4 Algorithm Design

5 Requirements Traceability

A Appendix

A.1 Used tools

A.2 Hours of work

This is the time spent redacting the DD

- Lorenzo Binosi - 0 hours

References

[1] Software Engineering 2 Project, AA 2016/2017 - *Project goal, schedule and rules*

[2] Software Engineering 2 Project, AA 2016/2017 - *Assignments 2*

[3] *Software Engineering 2: PowerEnJoy Requirements Analysis and Specification Document* Binosi Lorenzo Politecnico di Milano