Accademic Year 2016/2017

# POWER ENJOY

### DD Presentation Politecnico di Milano Software Engineering 2

Simone Bruzzechesse

Luca Franceschetti

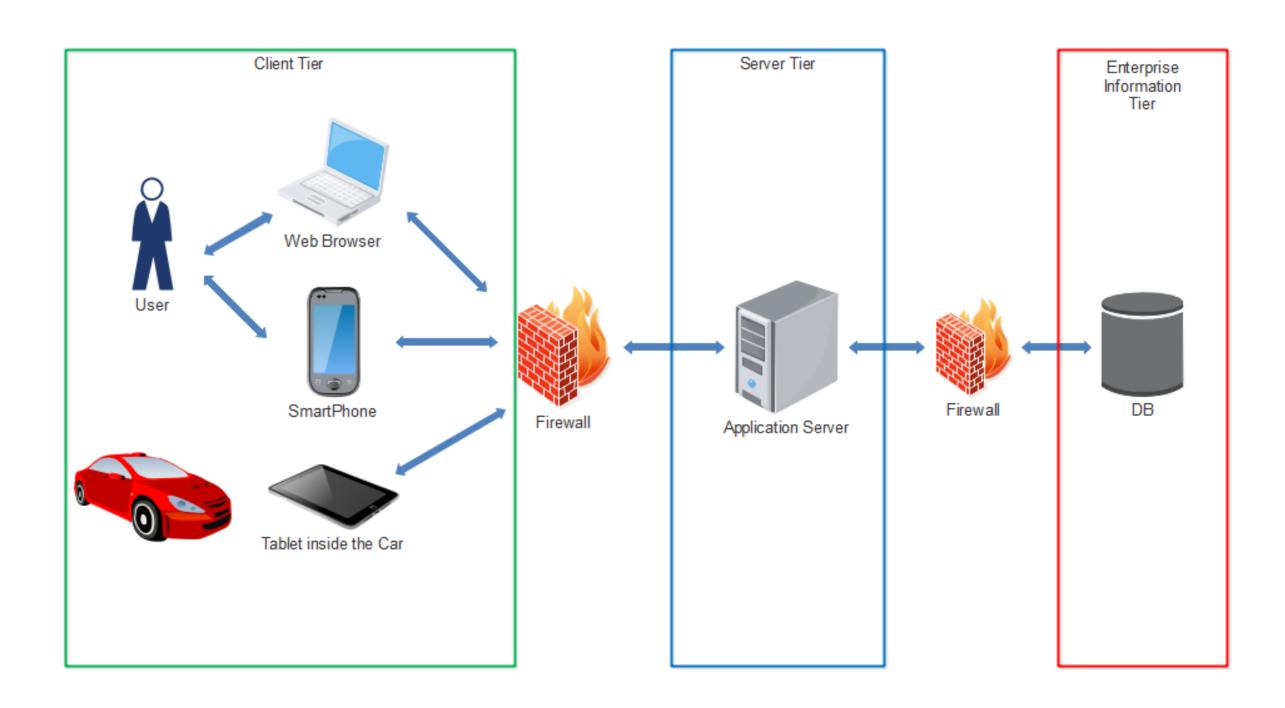
Gian Giacomo Gatti

#### **INDEX**

The purpose of Design Document is to identify:

- ➤ High level infrastructure
- ➤ Main components and their interfaces:
  - Component Diagram
  - ➤ Data Structure
  - Deployment View
- ➤ Runtime View
  - Sequence Diagram
  - Algorithm Design
- ➤ User Interface Design

#### HIGH LEVEL INFRASTRUCTURE

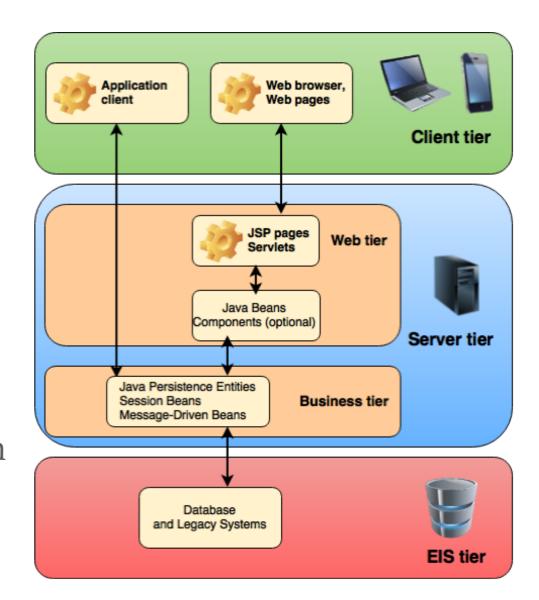


#### J2EE INFRASTRUCTURE

➤ Client-tier: run on the client machine, both on the web browser and mobile application and also on the machine's tablet

#### > Server-tier:

- ➤ **Web-tier:** Servlets and JSP that are used to manage the interaction with the web application
- ➤ **Business-tier**: these components manage the internal logic of the system
- ➤ Enterprise Information System (EIS)-tie: handle enterprise infrastructure systems, database systems and legacy systems



#### MAIN COMPONENT DIAGRAM

<<component>> CarApplication 包 <<component>> **NotificationService** CarManager 割 包 <<component>> <<component>> <<component>> Server **PaymentService** UserApplication UserActionManager ExternalServiceManager 包 <<component>> MapsService DatabaseManager <<component>> DBMS

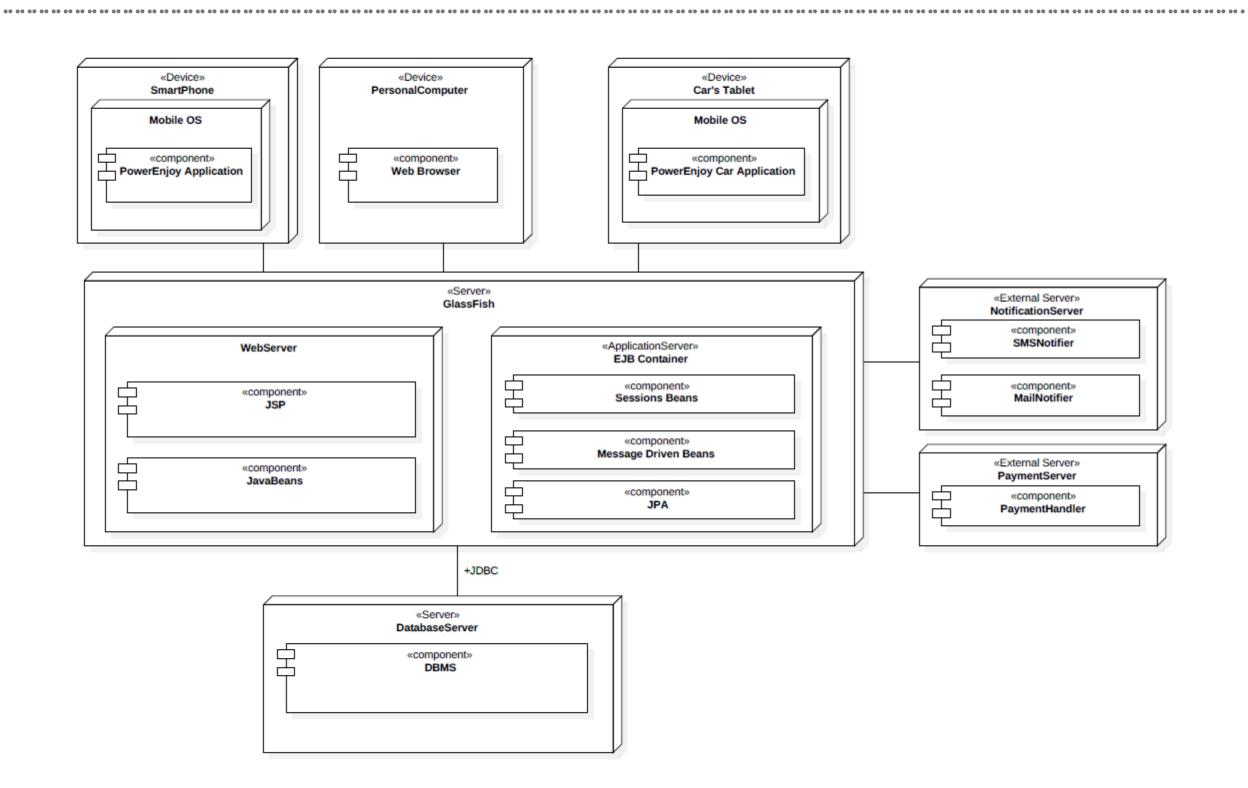
#### **COMPONENT DIAGRAM**

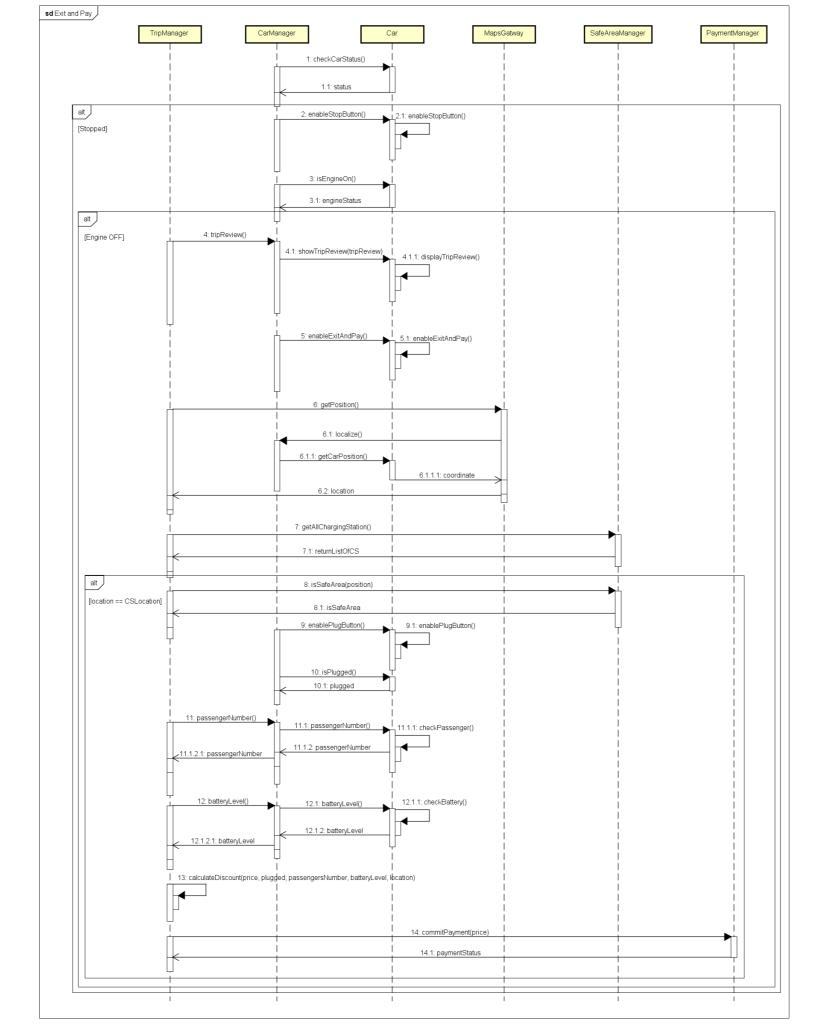
ClientTler <<component>> <<component>> <component>> Guest User ServerTier Registration Login CarInterface <<component>> <<component>> LoginManager <<component>> SecurityManager <<component>> <<component>> RegistrationManager Access Control NotificationGateway NotificationManage Λ NotificationGateway <<component>> <<component>> <<component>> CarManager Notification UserActionsManager ProfileManagement Profile Trip Reservation <<component>> <<component>> <<component>> <<component>> <<component>> PaymentGateway ReservationManager SafeAreasManager PaymentManager TripManager Payment Gateway Payment <<component>> DataBaseManager <<component>> <<component>> MapsGateway PostionManager Position **PositionGateway JDBS** DataBase Tier <<component>> DB

#### DATA STRUCTURE

If a safe area is a chargingStationArea than it will be associated to a chargingStation and the latitude and longitude will be the same of the Reservation isAssociatedTo isAssociatedTo station while the range will be of 3Km, +reservationCode: Int otherwise it won't be associated to a station. 0..\* +userCode: Int +carCode: Int 0..\* +time: Date +reservationState: ReservationState SafeAreas +latitude: Float +longitude: Float +range: Int +isChargingStationArea: Boolean 1 +chargingStationCode: Int Car User +carCode: Int +userCode: Int isAssociatedTo +plate: String +password: String +battery: Int +username: String +carState: CarState itCouldBeAssociated +userState: UserState 0..1 0..1 ChargingStation Trip 1 +NumberOfPlugs: Int +tripCode: Int +NumberOfCars: Int +userCode: Int +stationLatitude: Float +carCode: Int isAssociatedTo +stationLongitude: Float +reservationCode: Int isAssociatedTo +chargingStationCode: Int +startLatitude: Float +startLongitude: Float 0..\* +endLongitude: Float +endLatitude: Float +cost: Float Discount 0..\* +passengers: Int +discountCode: Int +discounts: Set +percentage: Int isAssociatedToANumberOf +startTime: Date +increase: Boolean +endTime: Date +tripState: TripState

#### **DEPLOYMENT VIEW**

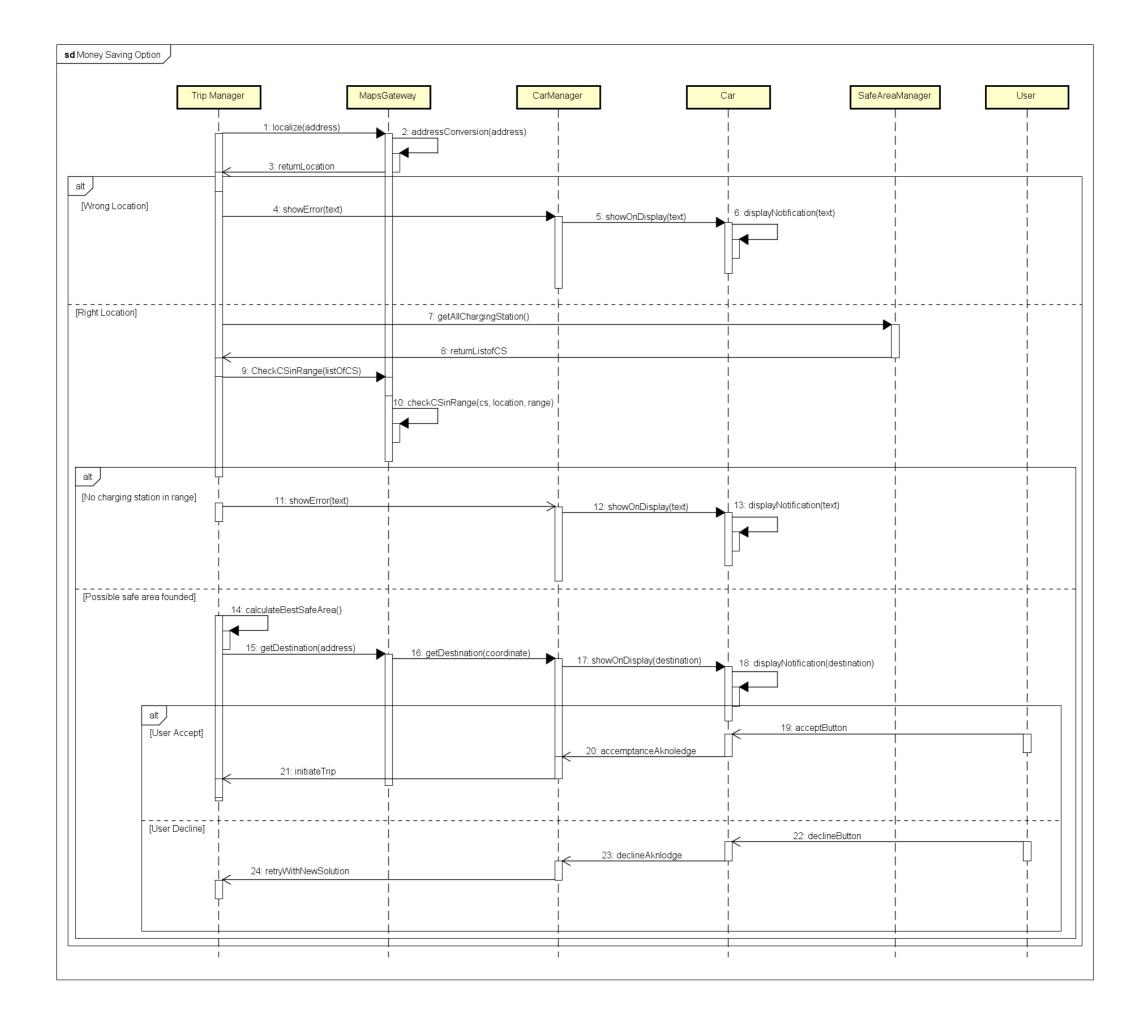




#### ALGORITHM DESIGN - EXIT AND PAY

This algorithm has the aim to explain the conclusion of the rent:

- √ There are a lot of possible alternatives at the end of the rent
- ✓ We should check more details in order to guarantee coherence between the **trip** and the **payment**
- ✓ The algorithm have to check the **status** of the car (if it stopped or not)
- ✓ The algorithm have to check if the user asks to **stop** its trip
- ✓ As soon as the engine is **stopped**, the system stops **charging** the user
- ✓ Car's tablet shows the **review** and the details of the trip
- ✓ The user can now touch the "Exit and Pay" button on the display
- ✓ The algorithm saves **current location** of the car thanks to an interaction between CarManager and MapsGateway
- √ The system downloads all the charging station (justified as update)
- ✓ CarManager enables the possibility to plug the car as well as enables "plug" button only if the users stops near a charging station
- ✓ TripManager calculates right discount of the trip
- ✓ PaymentManager tries to commit the payment, and the system must check if it successful



#### **ALGORITHM DESIGN - MONEY SAVING OPTION**

This algorithm manages the "Money Saving" option:

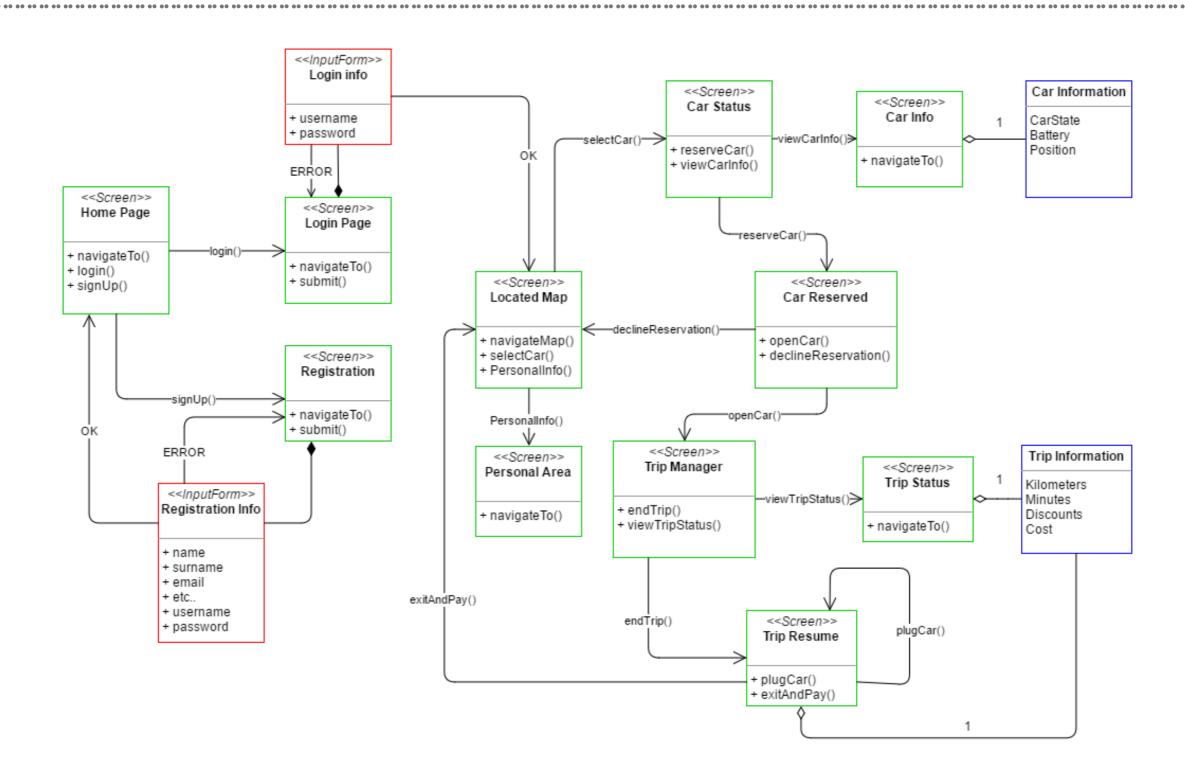
- ✓ We created a specific class for the location, to provide coherent and consistent information about car's and charging station's position.
- ✓ This function receives an address, the one that user has typed on the car's display, and a range, also typed on display.
- ✓ **CarManager** permits to interact with the display and consequentially with the user in the car.
- ✓ We created a specific class for the charging station, which are provided by SafeAreaManager during the computation.
- ✓ **MapsGateway** is a third part service and can calculate if a specific charging station is or not in range, from the location desired by the user.
- ✓ Before the user can see the result on display, the system **checks** which charging station **has less car** and advices its position to the user.

#### **USER INTERFACE DESIGN**

<<InputForm>> Login info + username + password ΟK ERROR <<Screen>> <<Screen>> Home Page Login Page -login()-+ navigateTo() + navigateTo() + login() + submit() <<Screen>> + signUp() Main Menu + navigateTo() + PersonalInfo() <<Screen>> Registration -signUp()-+ navigateTo() PersonalInfo() + submit() ΟK ERROR <<Screen>> Personal Area <<InputForm>> Registration Info + navigateTo() + name + surname + email + etc.. + username

+ password

#### **USER INTERFACE DESIGN**



## THANKS FOR YOUR ATTENTION