Power EnJoy

Electric car sharing system

01/03/2017

Francesco Peverelli - Federico Reppucci

Results of requirements analysis

Actors involved

PERSON

A person not registered to Power EnJoy or not logged into the system

EMPLOYEE

An employee responsible for the car retrieval

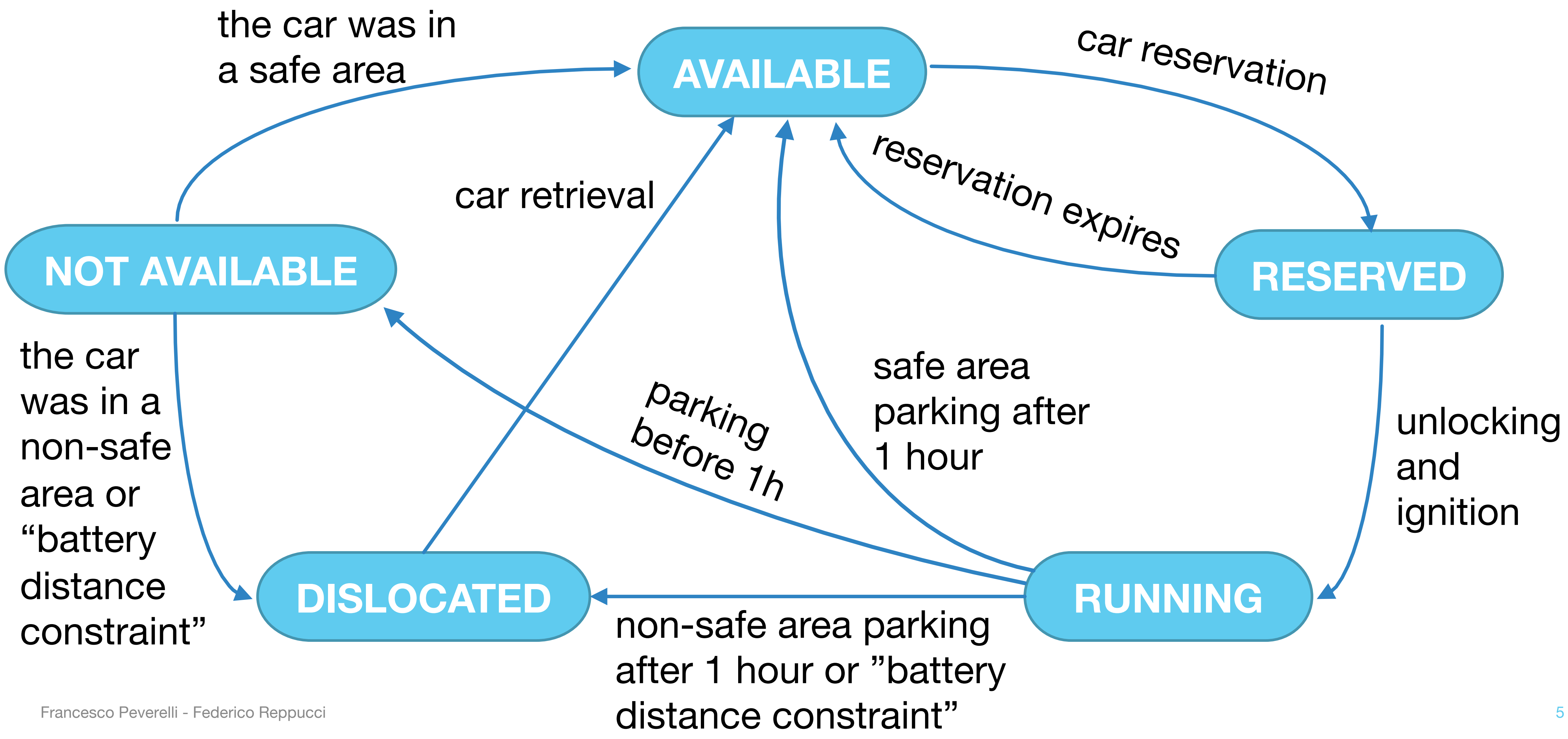
USERS

A person registered to Power EnJoy and logged in

Relevant concepts

POWER GRID STATION	Electric power supplier for recharging the cars
POWER ENJOY CAR	Electric powered vehicles owned by PowerEnJoy.
SAFE AREA	Public car parks, specific areas of the city and private PowerEnJoy car parks

States of a car


the car was in a safe area
AVAILABLE
car reservation
car retrieval
reservation expires
NOT AVAILABLE
RESERVED
the car was in a non-safe area or "battery distance constraint"
parking before 1h
safe area parking after 1 hour
unlocking and ignition
DISLOCATED
RUNNING
non-safe area parking after 1 hour or "battery distance constraint"

Car retrieval

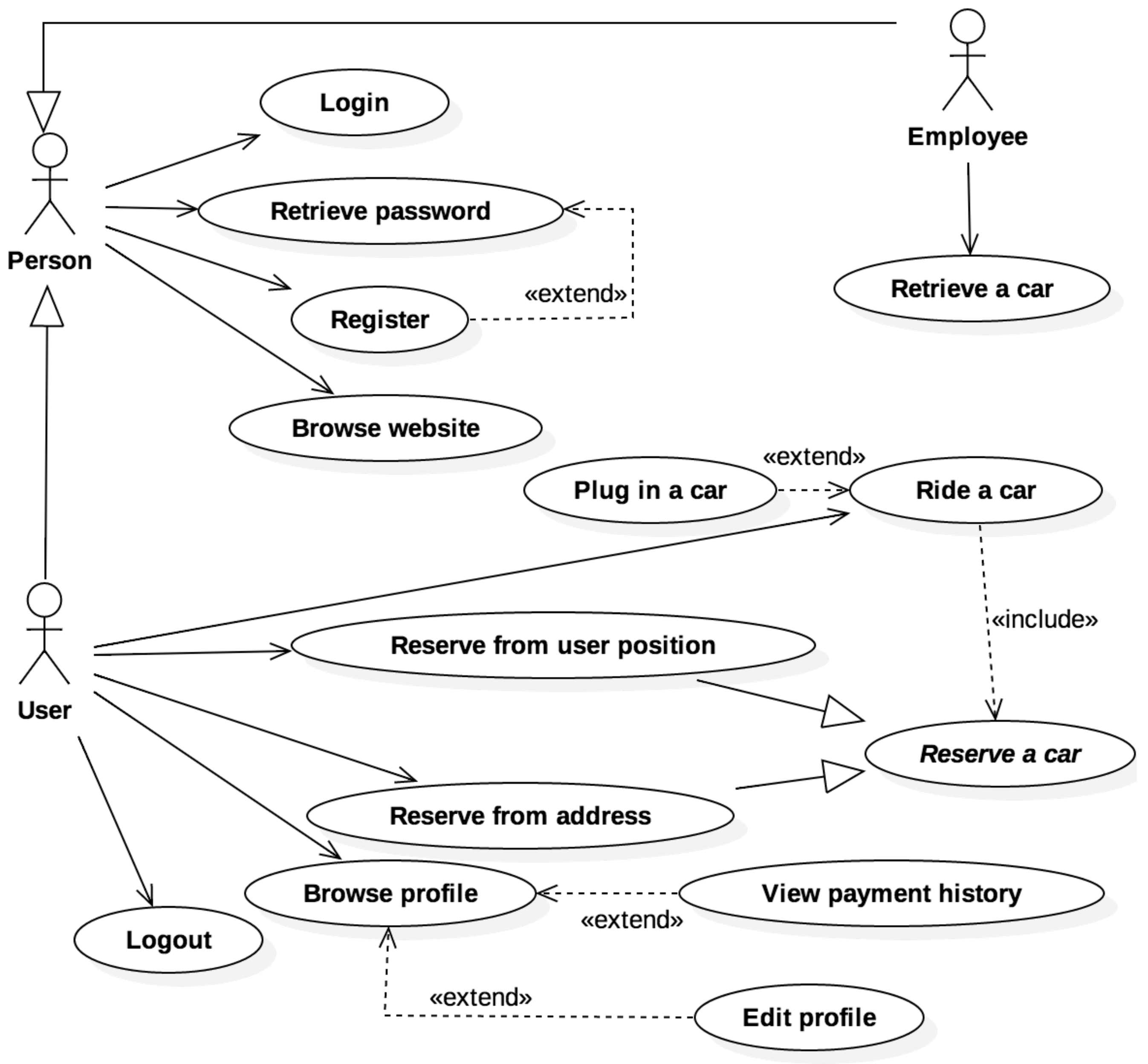
What if a user parks in a non-safe area?

- a set of employees has access to a dedicated application
- when a car enters the "dislocated" state, a notification is broadcast containing the specifics for the retrieval
- an employee accepts the retrieval request
- he/she brings the car back to a safe area, recharging it if needed

Extra fees and discounts

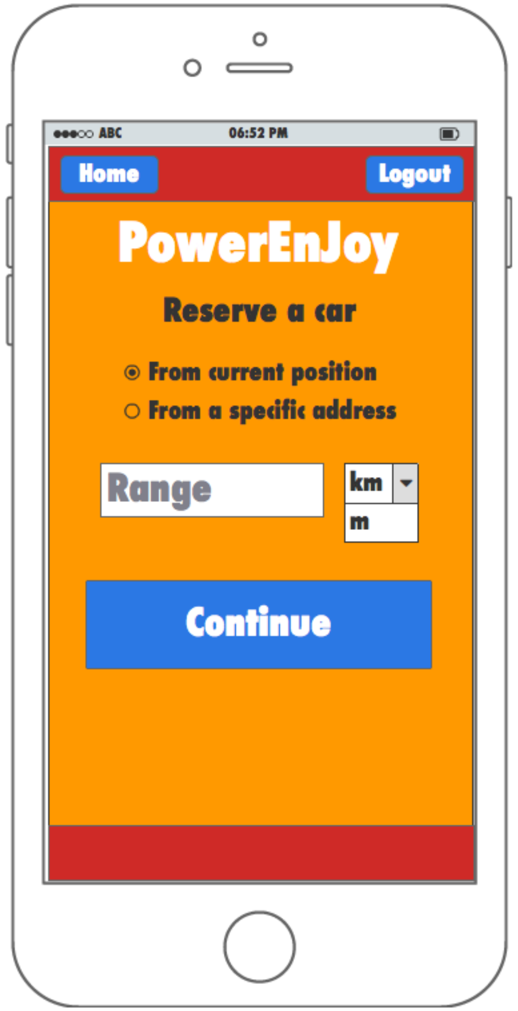
- Only the highest percentage discount is applied
- No discounts is applied if any extra fee is charged
- Extra charges are applied cumulatively

RATIONALE: we want to discourage negative behavior from the users and to prevent the users from abusing the discount system



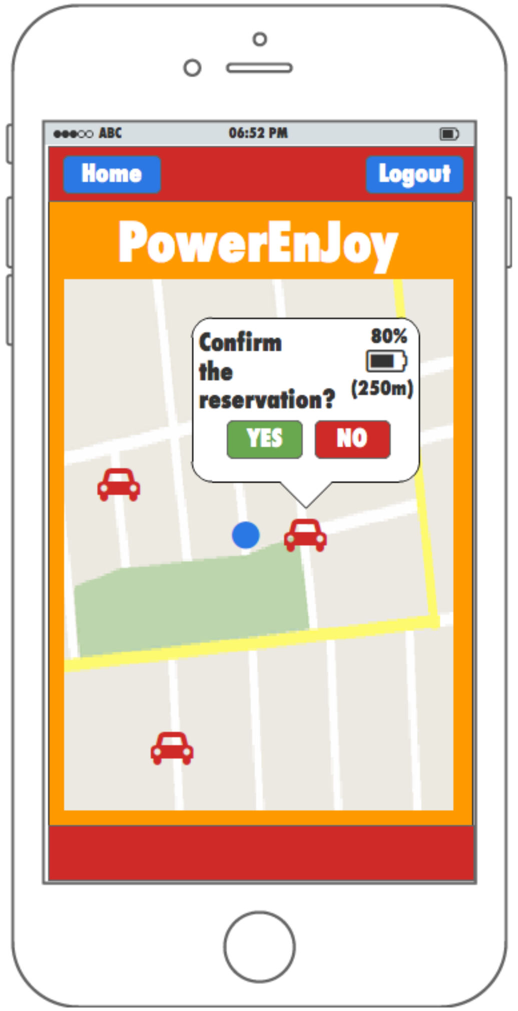
Functionalities overview

Look and feel



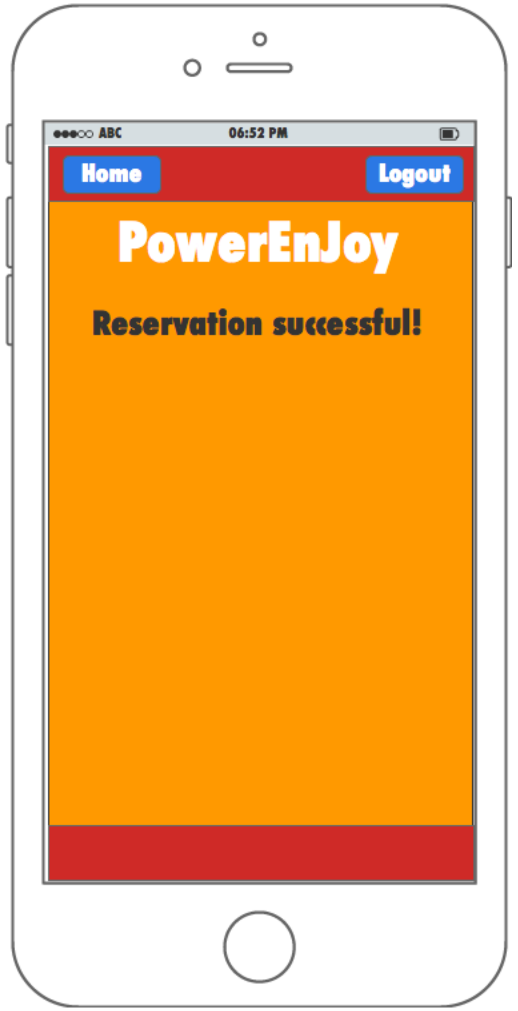
Reserve a car

[✓ From current position]
From a specific address

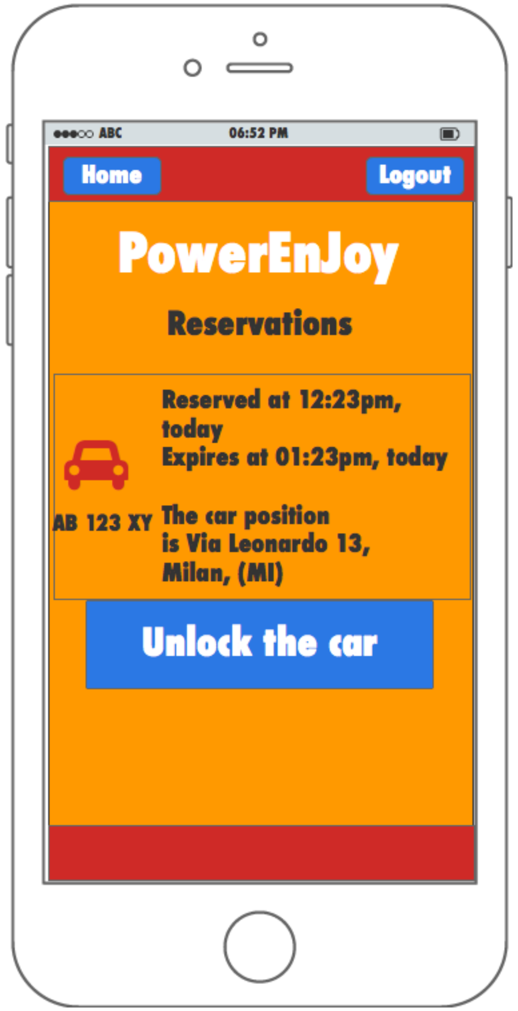


Confirm the reservation?

80%
(250m)



Reservation successful!



Reservations

Reserved at 12:23pm, today
Expires at 01:23pm, today

AB 123 XY The car position is Via Leonardo 13, Milan, (MI)

Let’s talk about software design…

Car software system

User application

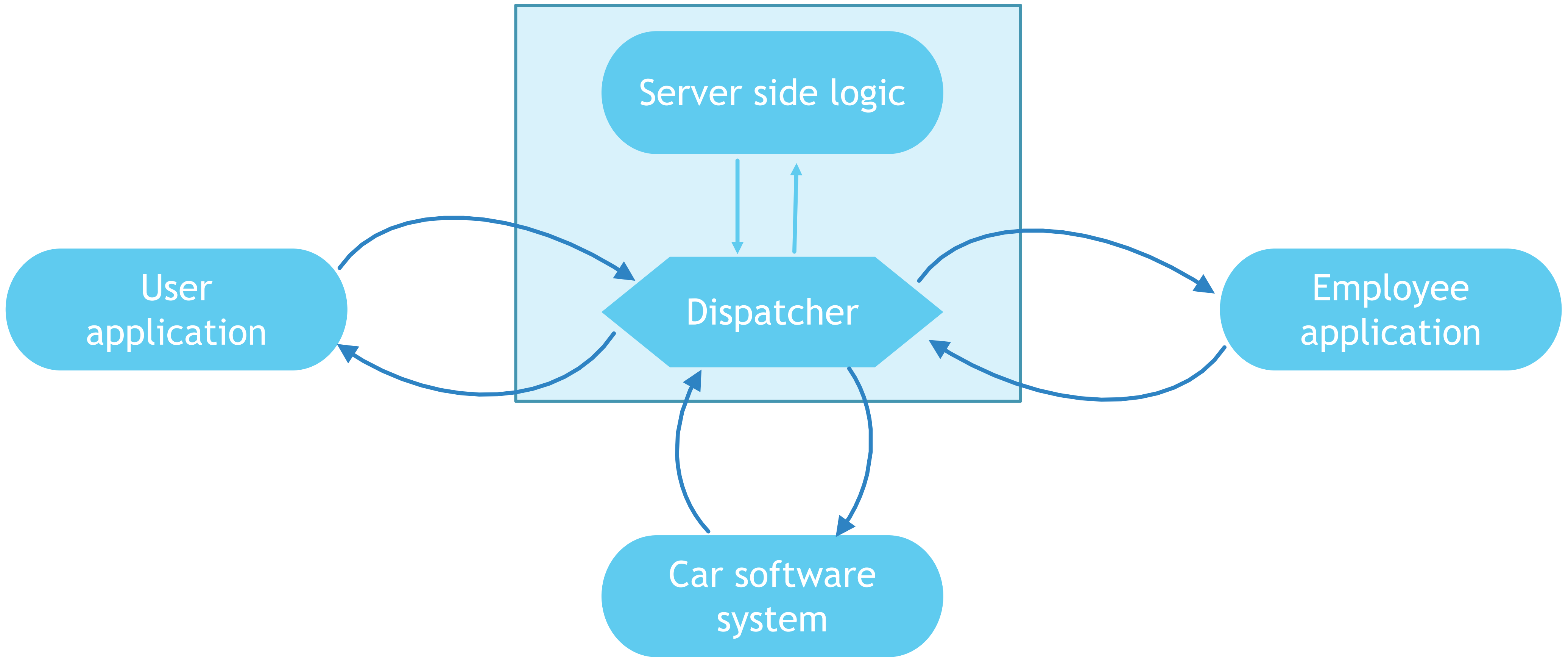
Employee application

Server side system

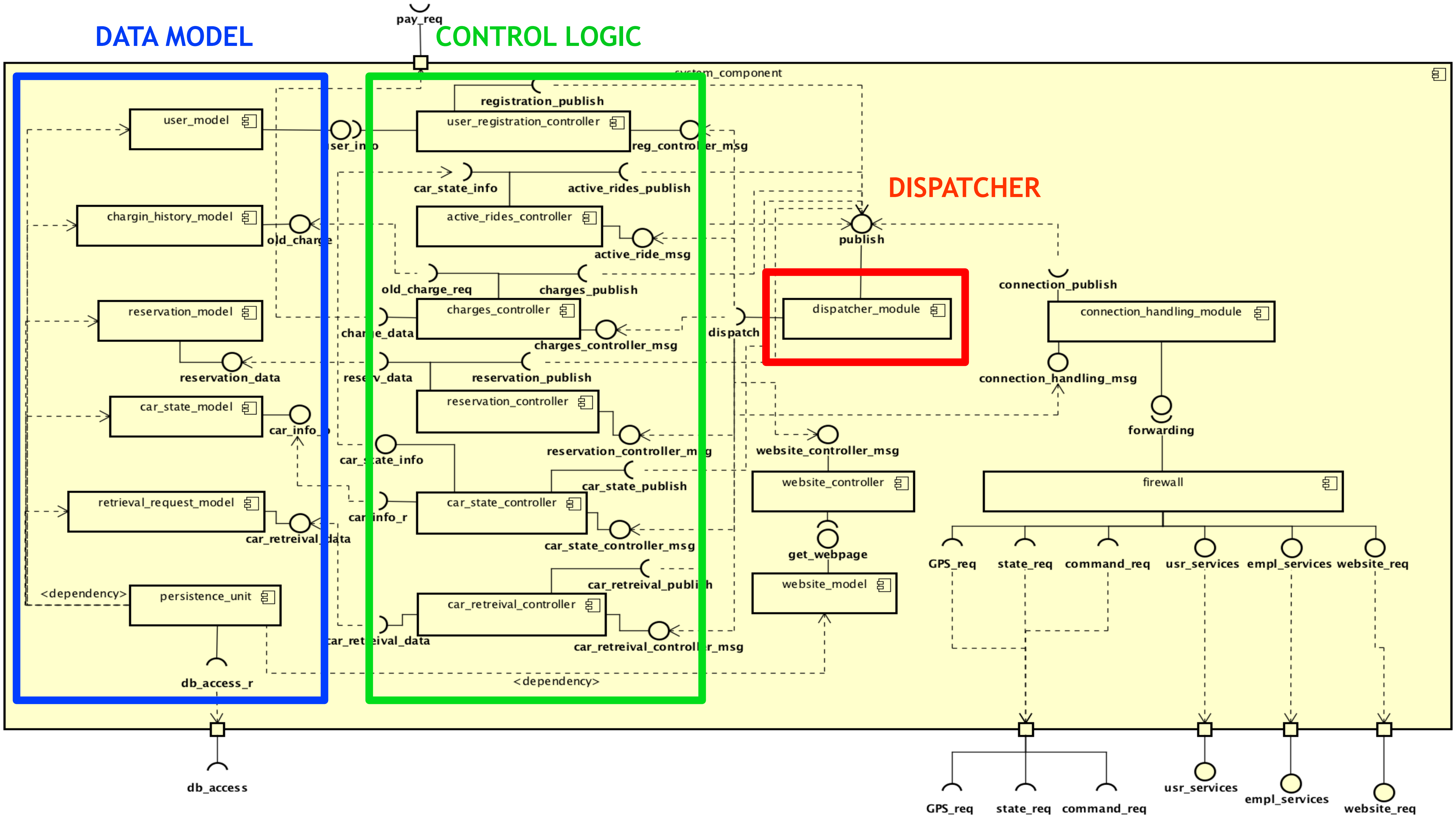
What does our software system need?

- Different software components need to exchange information
- Their interaction is largely **event-centric**:
  - Notifications about dislocated cars
  - State-change notifications from cars to the server side
  - Notification to the user about his/her ride

Event driven architecture

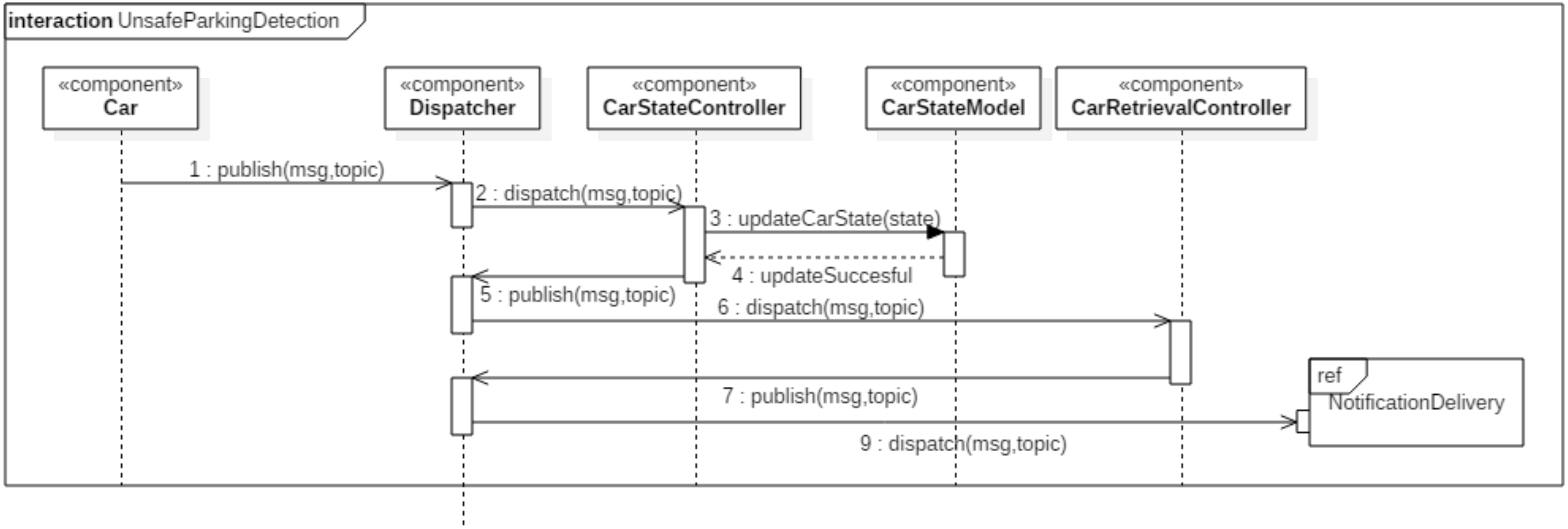


A closer look at the server side


DATA MODEL
CONTROL LOGIC
DISPATCHER
pay_req
system_component
user_model
chargin_history_model
reservation_model
car_state_model
retrieval_request_model
persistence_unit
<dependency>
db_access_r
db_access
user_info
old_charge
reservation_data
car_info_p
car_retreival_data
registration_publish
user_registration_controller
reg_controller_msg
car_state_info
active_rides_publish
active_rides_controller
active_ride_msg
old_charge_req
charges_publish
charges_controller
charge_data
charges_controller_msg
reserv_data
reservation_publish
reservation_controller
reservation_controller_msg
car_state_info
car_state_publish
car_state_controller
car_info_r
car_state_controller_msg
car_retreival_publish
car_retreival_controller
car_retreival_data
car_retreival_controller_msg
<dependency>
publish
dispatch
dispatcher_module
connection_publish
connection_handling_module
connection_handling_msg
forwarding
website_controller_msg
website_controller
get_webpage
website_model
firewall
GPS_req
state_req
command_req
usr_services
empl_services
website_req
GPS_req
state_req
command_req
usr_services
empl_services
website_req

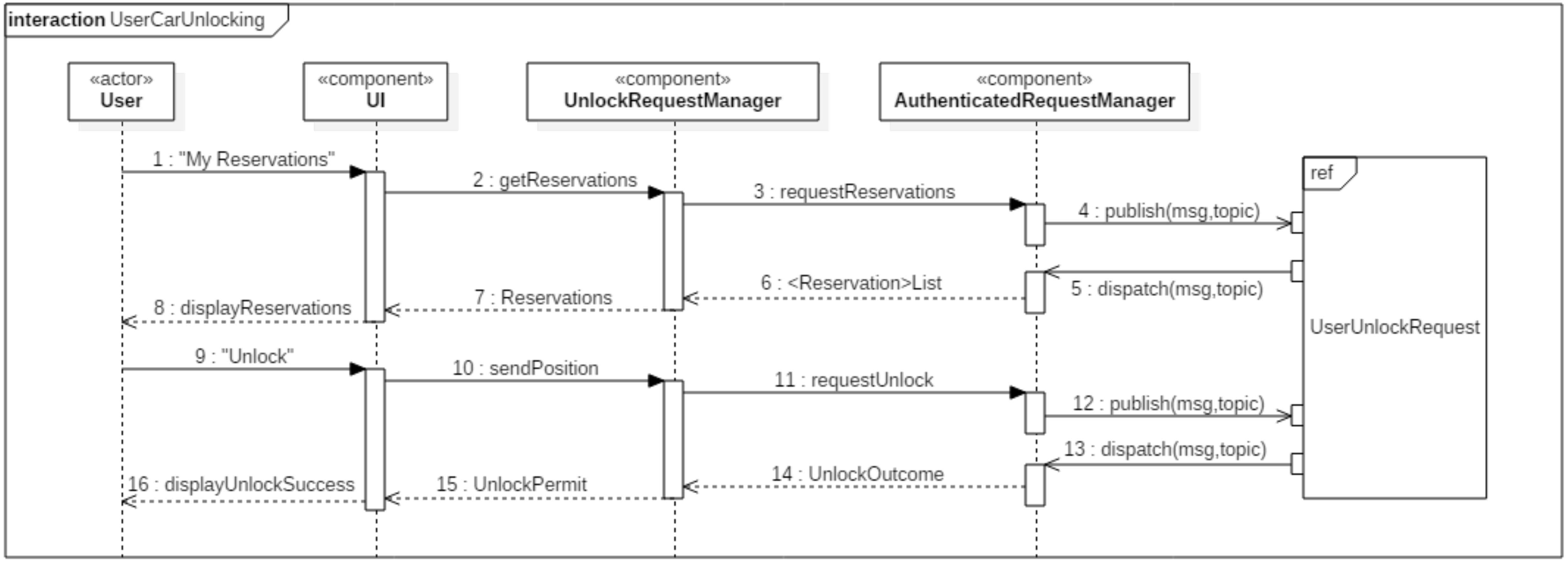
Interaction example

Unsafe parking detection



Interaction example

User car unlocking



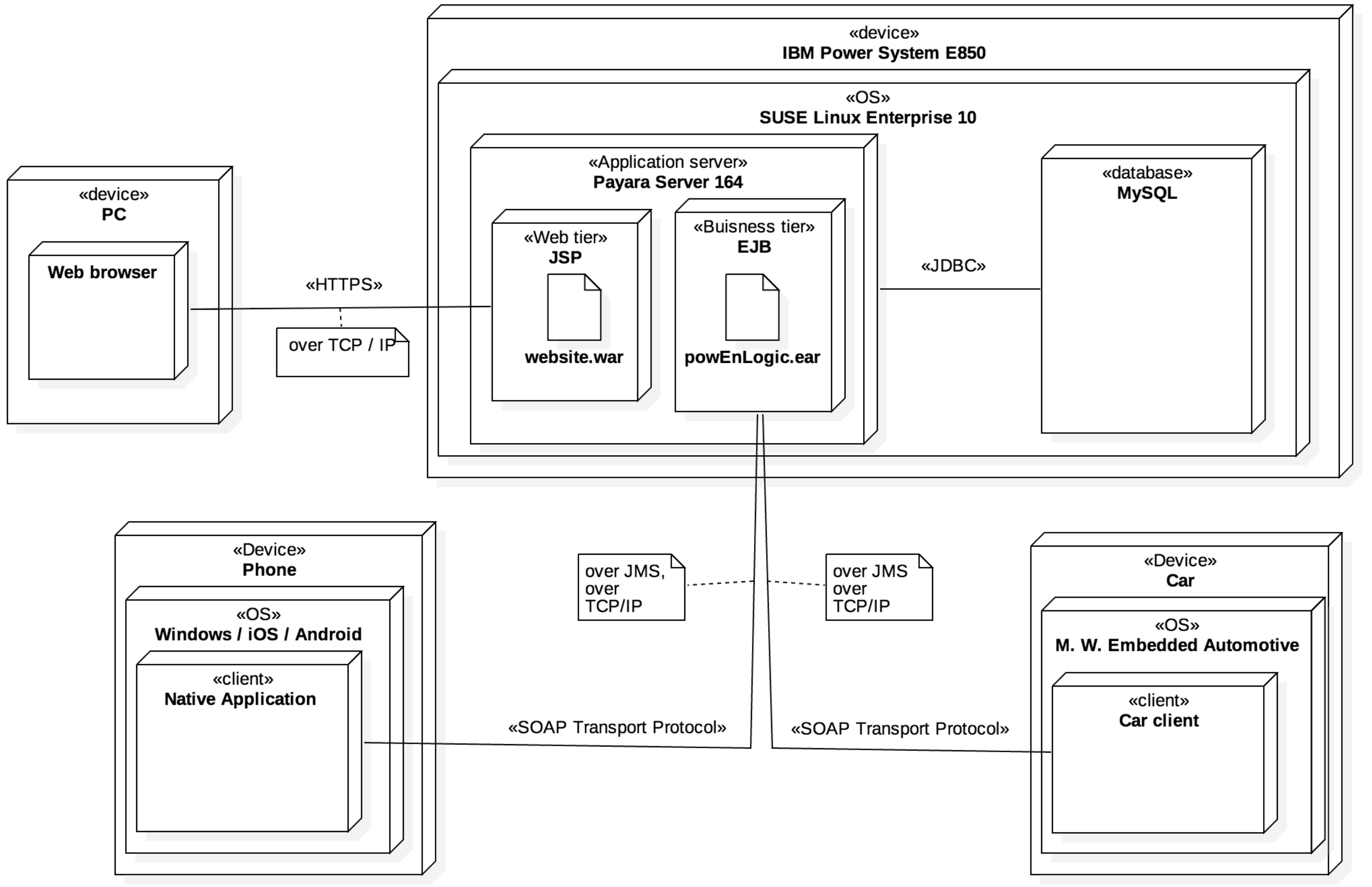
From design to implementation

- A stable and well known framework for enterprise applications development
- Supports event-driven architecture (Java Message Service)
- Helps to deal with scalability and availability issues

For a better user experience

Dedicated application for all the main operating systems

The resulting deployment



A few words on testing

For integration testing we adopted a **bottom-up** testing strategy

This will allow us to perform integration testing as soon as the involved components have been tested at the unit level, since the development process will also follow bottom-up approach

A **thread-based** strategy has been considered, but we opted out of it due to the multi-functional nature of many components

Thank you for the attention

Francesco Peverelli, Federico Reppucci