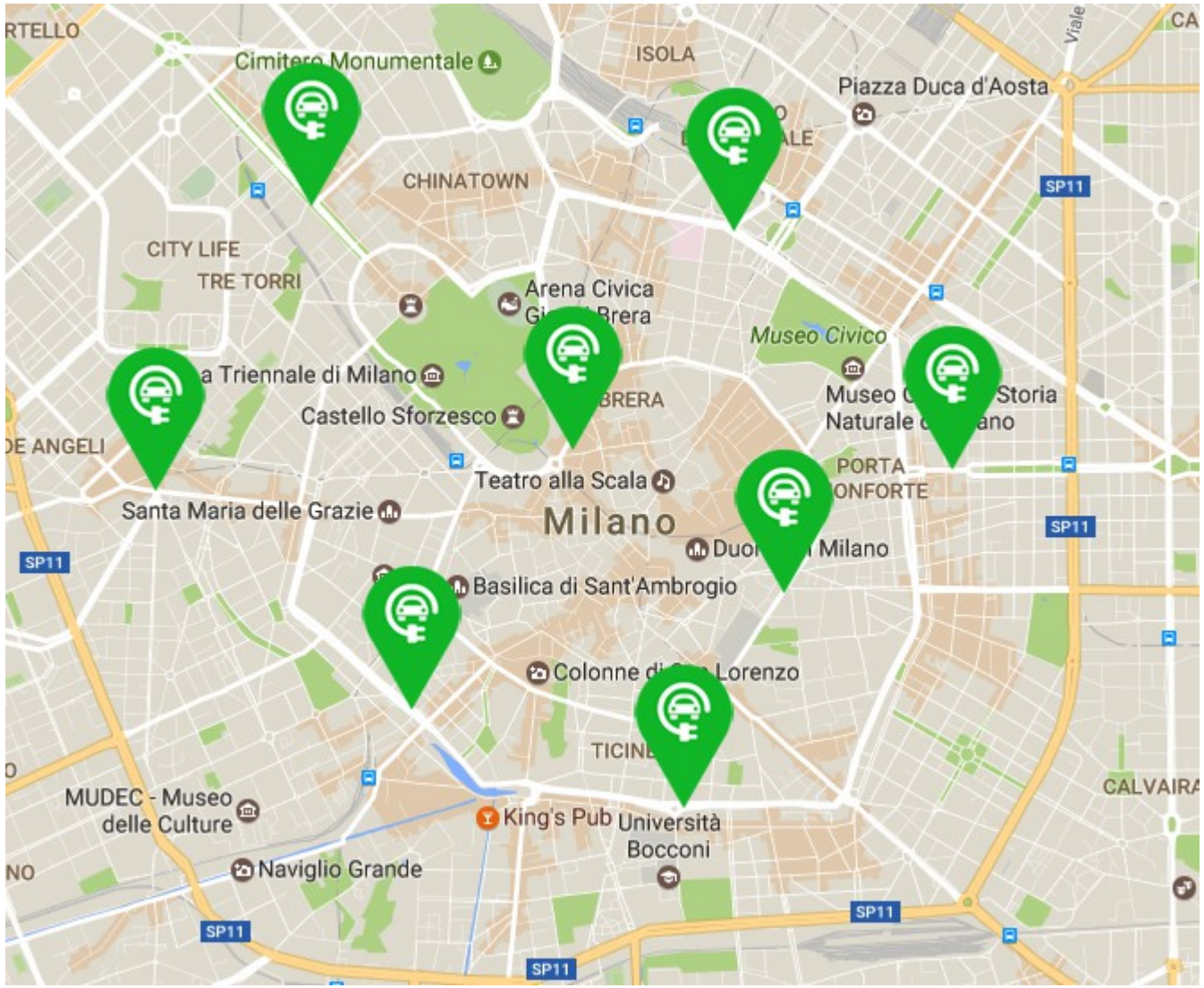
Software Engineering 2:

Politecnico di Milano

Lorenzo Binosi

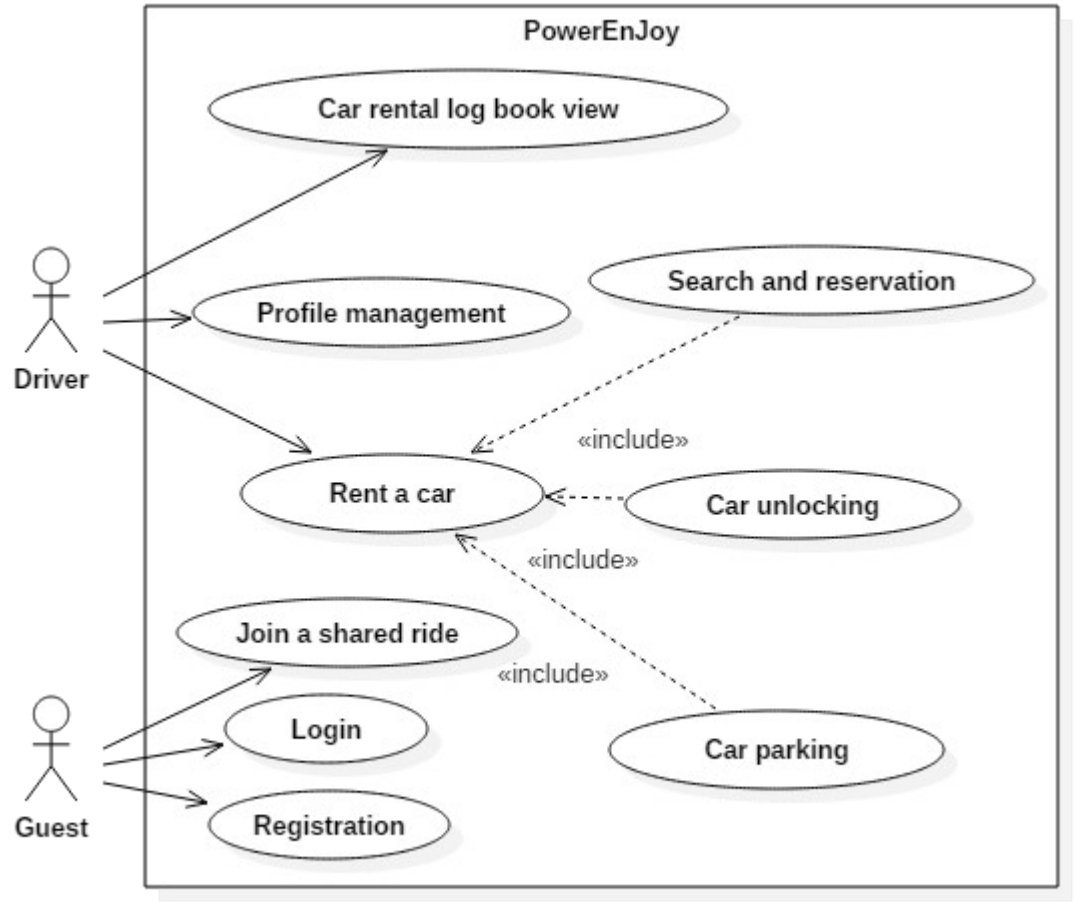
What is PowerEnJoy?



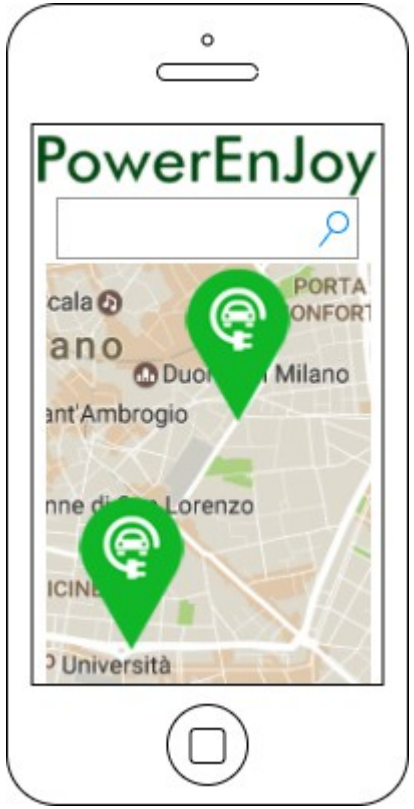
Goals

- [G1] Allow a driver to rent a car.
  - [G1.1] Allow a driver to search and reserve a car.
  - [G1.2] Allow a driver to use a reserved car.
  - [G1.3] Allow a driver to park a car in a safe area.
- [G2] Allow a guest to sign up to the system.
- [G3] Motivate a driver into car pooling.
- [G4] Motivate a driver to a better behavior while renting a car.

Use case diagram



User interfaces



Mobile application

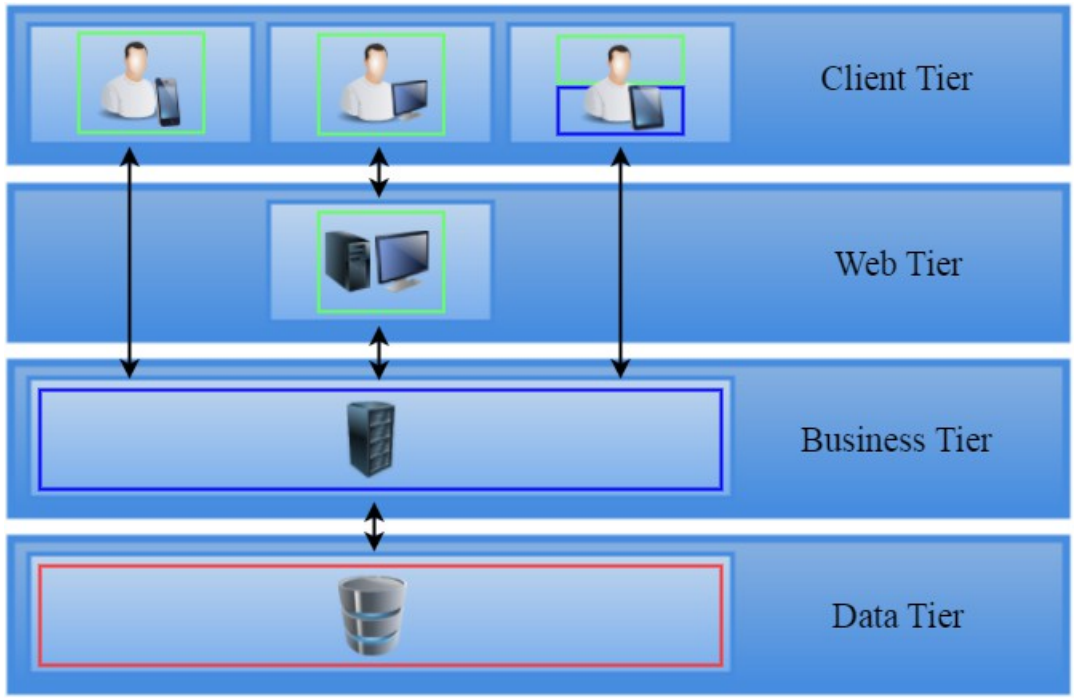
Web application

Assumptions

- All drivers have access to a stable internet connection.
- Once a driver unlocks a car, he will use it for his trip.
- Before leaving the car, the driver close it correctly.
- The system is operating in an area composed by a city or agglomerated cities, of the same country, closed each other.
- All the areas where the system is operating are covered by a reliable 3G/4G connection.

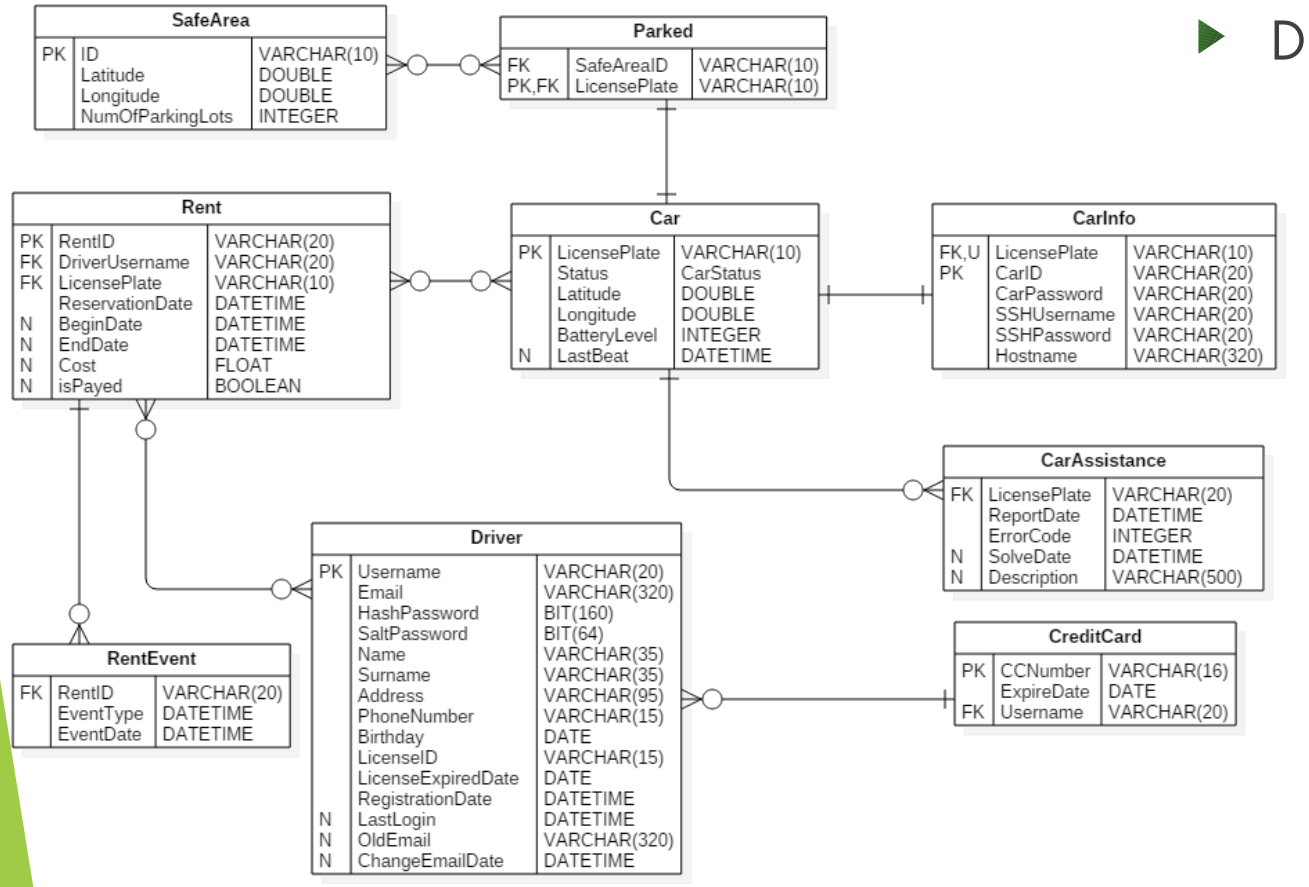
Architectural Design

Architecture overview



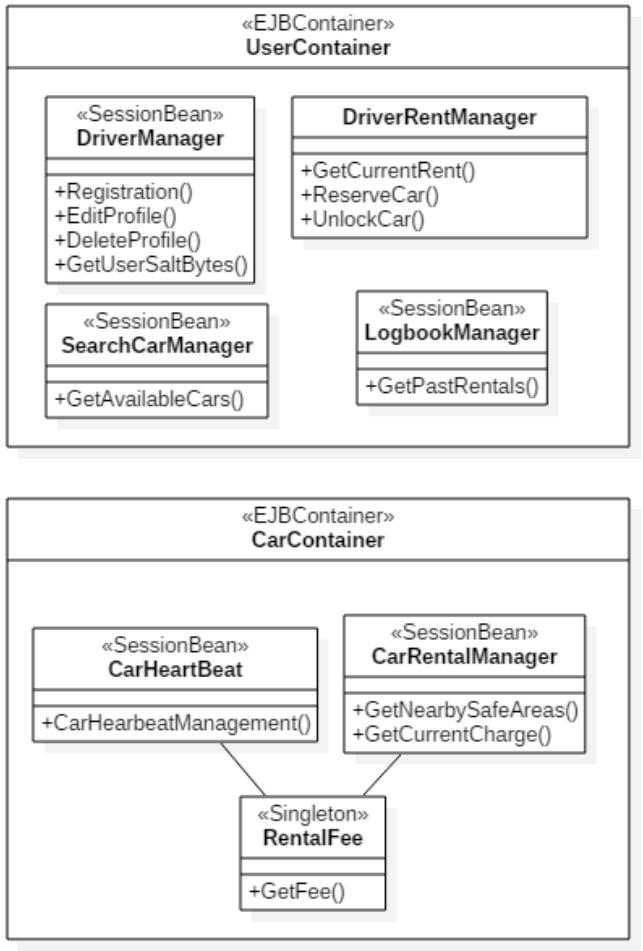
- Client tier:
  - Mobile Application
  - Web browser
  - Car Applications
- Web tier:
  - Web server
- Business tier:
  - Application server
- Data tier:
  - Database

High level components



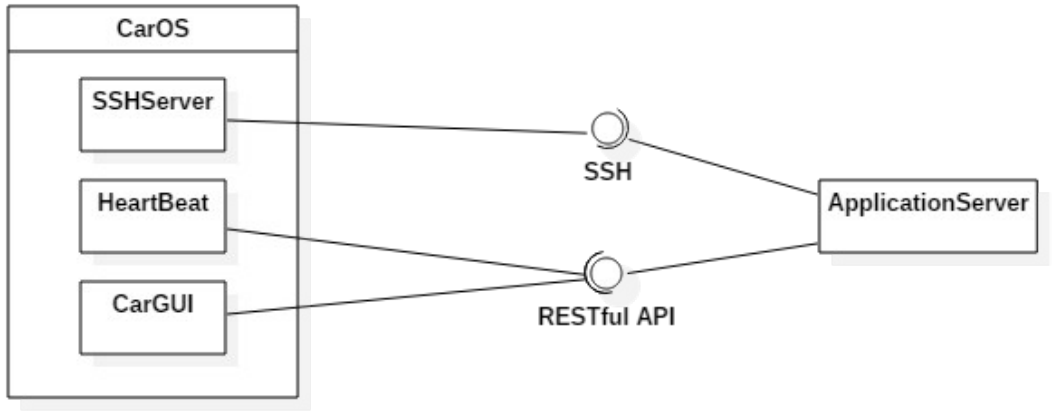
- Database:
  - Allows to store and retrieve all the persistent data
  - Runs MySQL Community Edition 5.7
  - Adheres to ACID properties with InnoDB

High level components



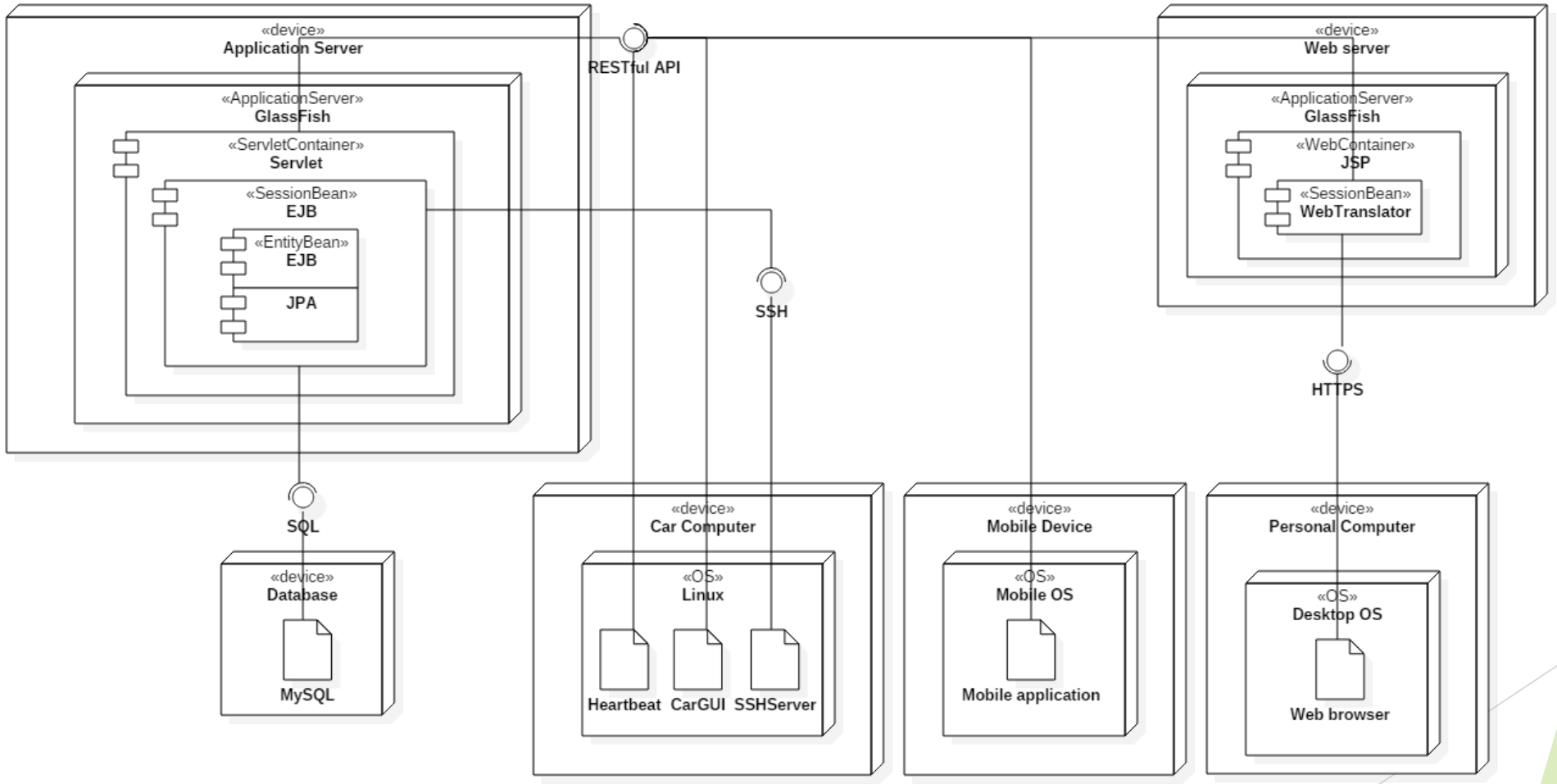
- Application server:
  - Contains the business logic of the system
  - Runs on a Glassfish server
  - Communicates with the database through JPA
  - Implemented by custom EJB
  - Provides RESTful API

High level components



- Car applications:
  - Send information to the application server in order to be tracked
  - Provide an SSH interface in order to be locked/unlocked remotly
  - Provide a GUI to the driver with utility functionalities

Deployment view



About REST

- REpresentational State Transfer
- Architectural style and approach to stateless communications
- Architectural properties:
  - Performance
  - Scalability
  - Simplicity
  - Modifiability
  - Reliablity

RESTful API

POST	/api/user/add-user	Creates user (REST)
GET	/api/user/{username}/salt-bytes	Retrieves user's salt bytes (REST)
DELETE	/api/user/delete-user	Deletes user from the system (REST)
GET	/api/search-cars/{latitude}/{longitude}	Retrieves nearby available cars (REST)
GET	/api/rental-operations/{licensePlate}/reserve	Reserves car (REST)
GET	/api/rental-operations/current-reservation	Retrieves current reservation (REST)
GET	/api/logbook/past-rents	Retrieves user's rental logbook (REST)
PUT	/api/rental-operations/unlock-reserved-car	Unlocks car (REST)
GET	/api/car/safe-areas/{latitude}/{longitude}	Retrieves nearbyavailable safe areas (REST)
PUT	/api/car/heart-beat	Car Heartbeat (REST)

RESTful API request example

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

Response Classes

Return Value: AvailableCars

AvailableCars – This object contains information about the available cars.

Name	Data Type	Description
cars	Array[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	number	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	number	The latitude value.
longitude	number	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	number	The latitude value of the user's position.
longitude	number	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.

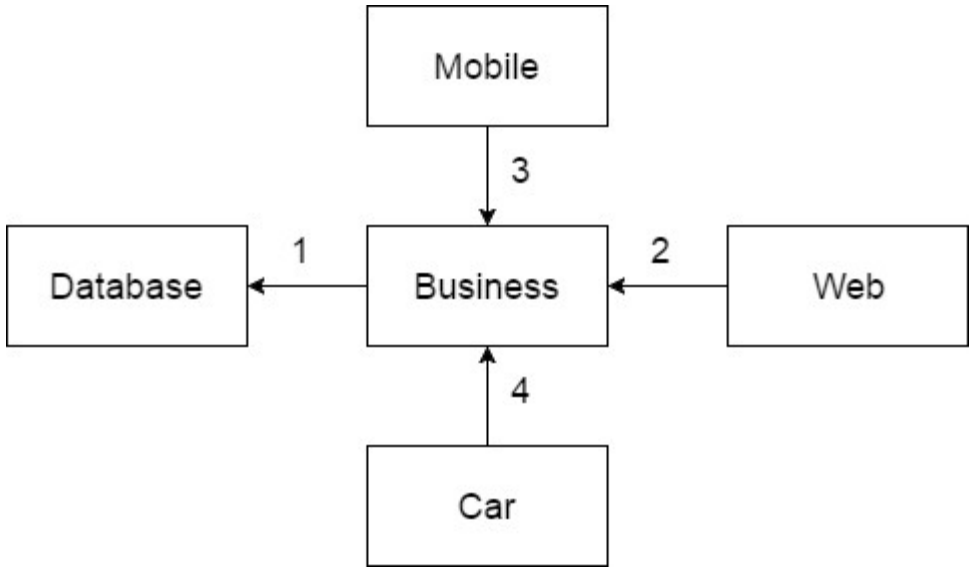
Integration testing

Entry criteria

- RASD and DD must be delivered and tested
- Once completed, components and subsystems can be integrated following the strategy adopted

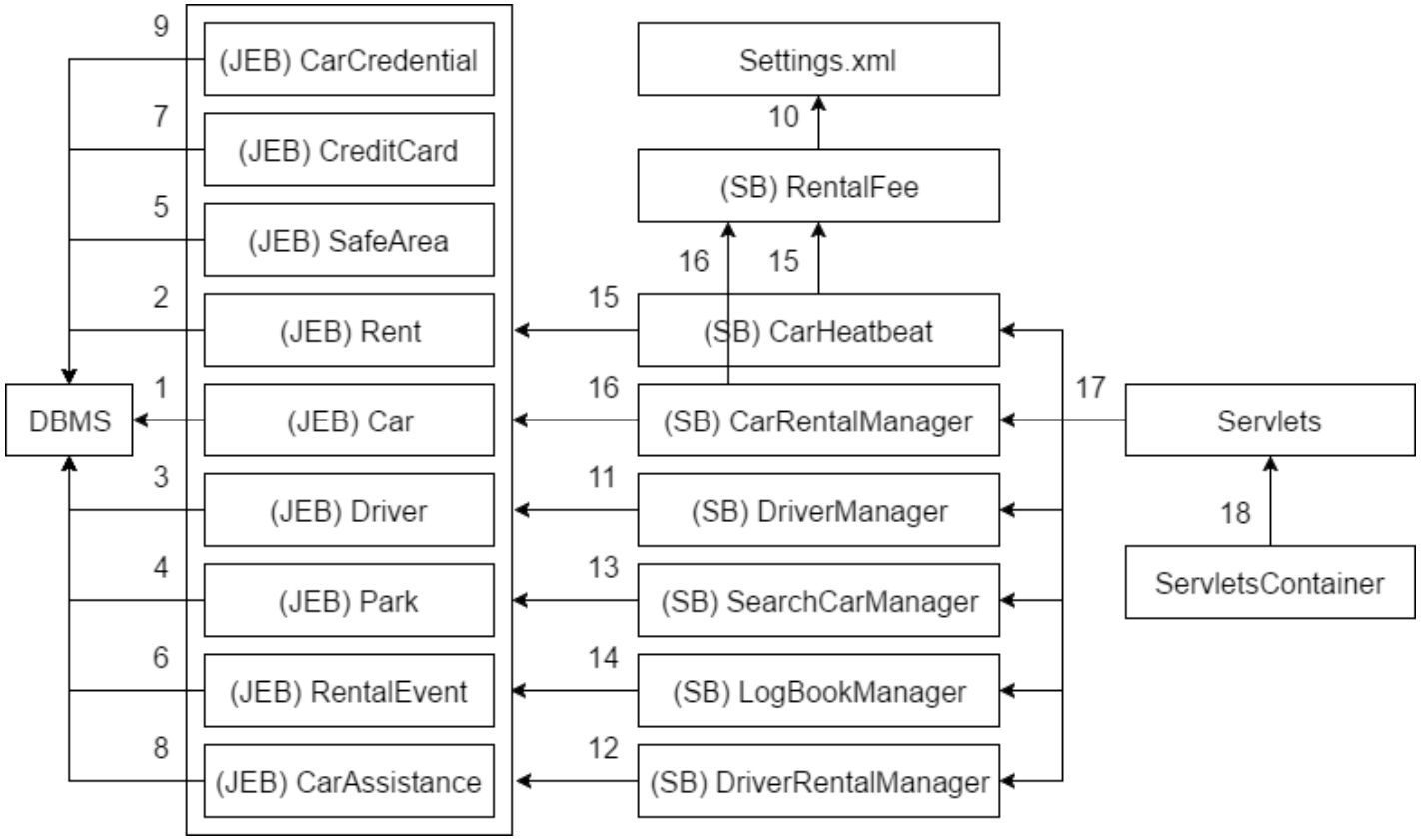
Subsystems strategy and sequence

- Critical-module-first approach

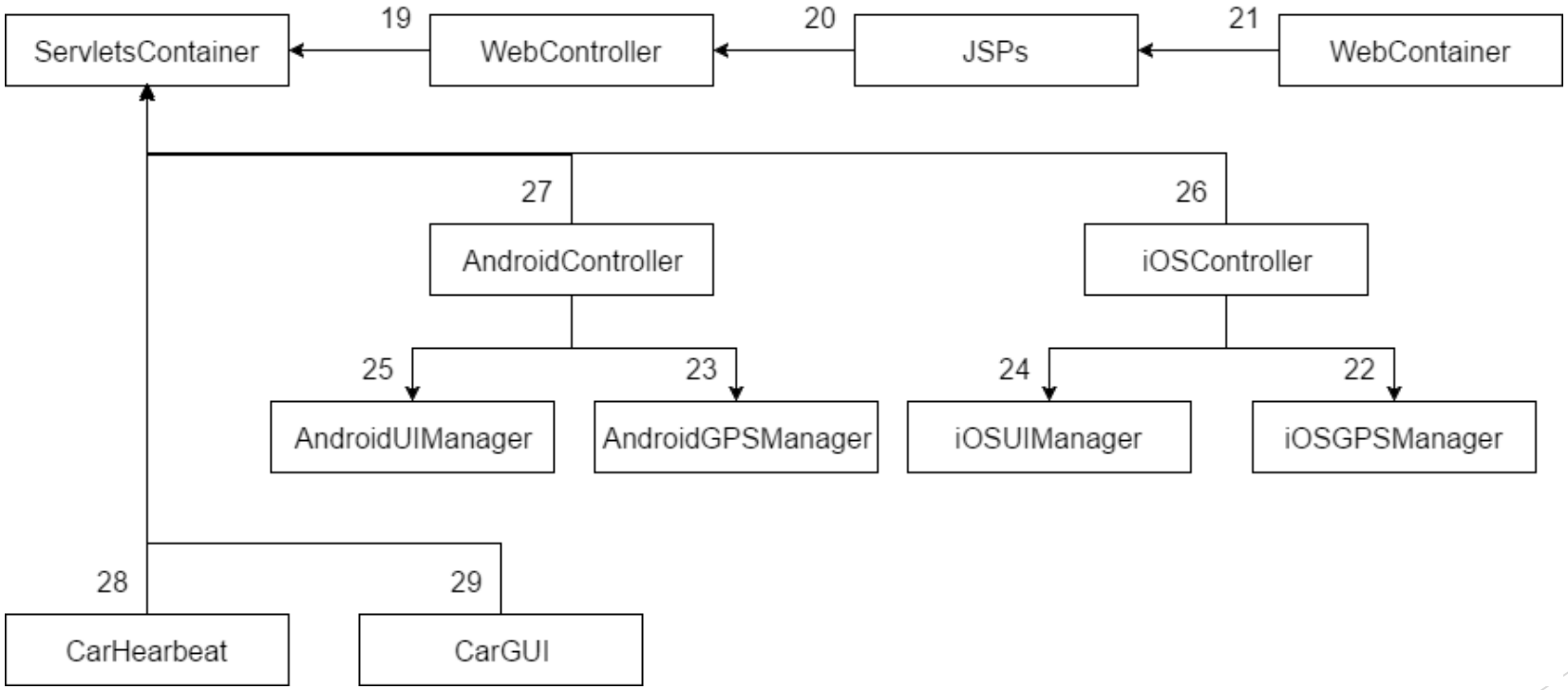


Components strategy and sequence

- Bottom-up approach



Components strategy and sequence



Size, cost and effort estimation

Project team

- Members: Lorenzo, Bob and Alice
- Programmers with average skills
- Members have never worked in other teams
- First project for the team

Function points

EI	Complexity	FPs
User creation	High	6
User's salt bytes retrieval	Low	3
User deletion	Low	3
User's rental logbook retrieval	Low	3
Available cars' retrieval	Low	3
Car reservation	Low	3
Current reservation retrieval	Low	3
Unlocks car	Low	3
Car heartbeat	High	6
Available safe areas' retrieval	Low	3
Total		**36**

EIF	Complexity	FPs
SMSGateway	Low	5
EmailSender	Low	5
PaymentGateway	Low	5
Total		**15**

EO	Complexity	FPs
Rent concluded	Low	4
Rent not concluded	Low	4
Payment received	Low	4
Payment not received	Low	4
Total		**16**

EQ	Complexity	FPs
User's salt bytes retrieval	Low	3
User's rental logbook retrieval	Low	3
Available cars' retrieval	Low	3
Current reservation retrieval	Low	3
Available safe areas' retrieval	Low	3
Total		**15**

ILF	Complexity	FPs
Driver	Low	7
Rent	Low	7
Car	Low	7
SafeArea	Low	7
CarAssistance	Low	7
Total		**35**

Function points

Function Type	Value
Internal Logic Files	35
External Interface Files	15
External Inputs	36
External Outputs	16
External Inquiries	15
Total	**117**

- Total amount of unadjusted function points = 117
- Java multiplicator = 53
- SLOC = 117 x 53 = 6201 source line of code

Scale factors

Code	Name	Factor	Value
PREC	Precedentedness	Low	4.96
FLEX	Development flexibility	Low	4.05
RESL	Risk resolution	Very High	1.41
TEAM	Team cohesion	Low	4.38
PMAT	Process maturity	Nominal	4.68
Total	$E = 0.91 + 0.01 \times \sum_i SF_i$		**1.1048**

Cost drivers

Code	Cost Driver	Factor	Value
RELY	Required software reliability	High	1.10
DATA	Database size	High	1.14
CPLX	Product complexity	Nominal	1.00
RUSE	Required reusability	Low	0.95
DOCU	Documentation match to life-cycle needs	Very High	1.23
TIME	Execution time constraint	Very High	1.29
STOR	Storage constraint	Extra High	1.46
PVOL	Platform volatility	Low	0.87
ACAP	Analyst capability	Nominal	1.00
PCAP	Programmer capability	Nominal	1.00
APEX	Application experience	Low	1.10
PLEX	Platform experience	Nominal	1.00
LTEX	Language and tool experience	Nominal	1.00
PCON	Personnel continuity	Low	1.12
TOOL	Usage of software tools	High	0.90
SITE	Multisite development	Very High	0.86
SCED	Required development schedul	Nominal	1.00
Total	$EAF = \prod_i C_i$		**2.2895**

Effort estimation

$$\text{Effort} = A \times EAF \times KSLOC^{E}$$

- Effort = 50.5 person-months, with
  - A = 2.94 (According to COCOMO II)
  - EAF = 2.2895 (Cost drivers)
  - E = 1.1048 (Scale factors)
  - KSLOC = 6.201

Duration

$$\text{Duration} = 3.67 \times \text{Effort}^{F}$$

- Duration = 12.82 months, with
  - Effort = 50.5 person-months
  - F = 0.28 + 0.2 x (E - B)
  - E = 1.1048 (Scale factors)
  - B = 0.91

People required

$$People = \lceil Effort / Duration \rceil$$

- People = 4, with
  - Effort = 50.5 person-months
  - Duration = 12.82 months

$$Duration = Effort / People$$

- Duration = 16,83 months, with
  - Effort = 50.5 person-months
  - People = 3 (since our team is composed by 3 people)

Thank You!