



PowerEnjoy

Software Engineering 2 Project AA 2016-17

Piantella Davide

Scrocca Mario

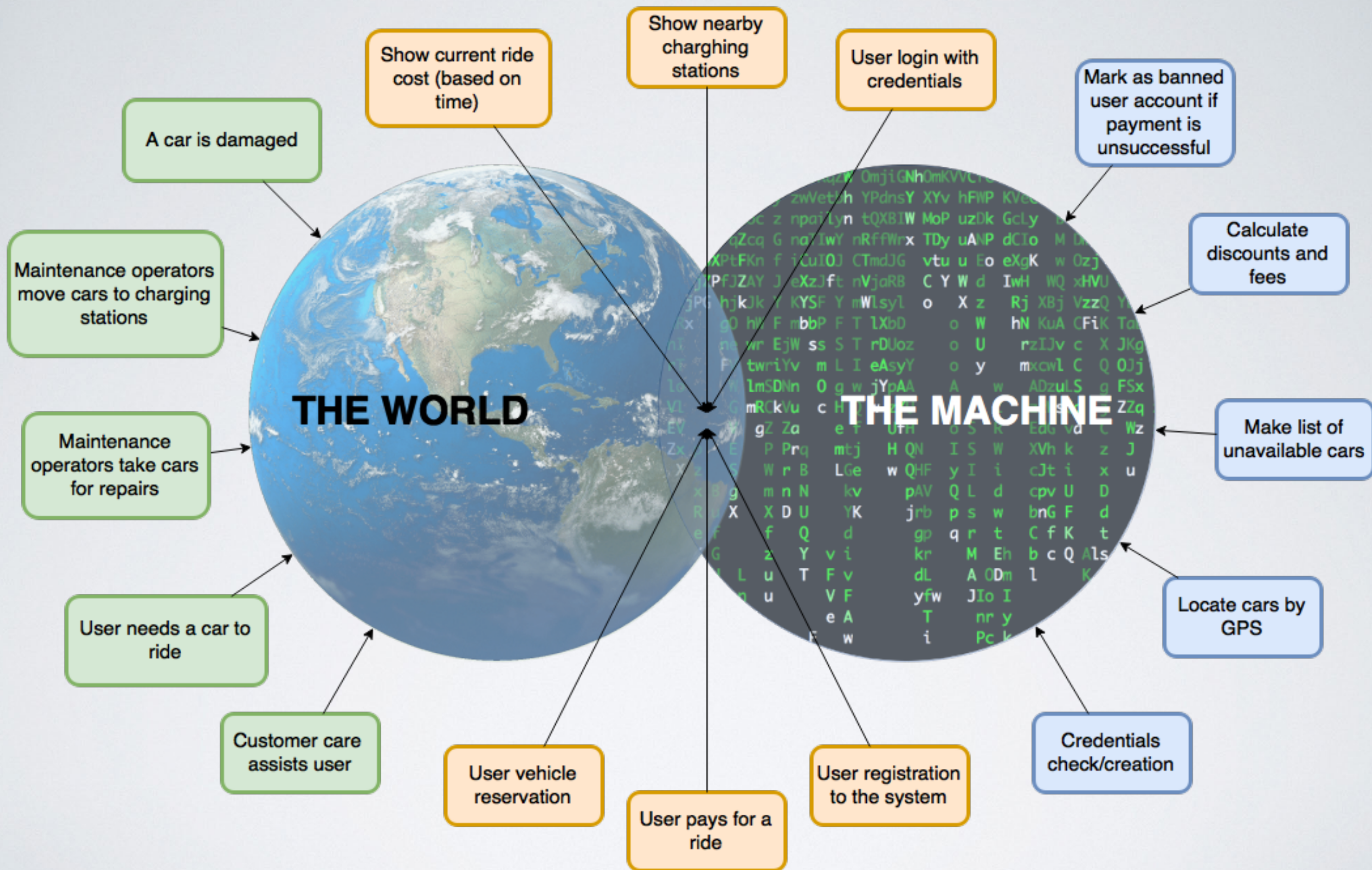
Vendra Moreno Raimondo

prof. Luca **Mottola**

PowerEnjoy

PowerEnjoy is a **car-sharing service** that exclusively employs electric cars; we are going to develop a web-based software system that will provide the functionalities normally provided by car-sharing services, such as allowing the user to register to the system in order to access it, showing the cars available near a given location and allowing a user to reserve a car before picking it up.

REQUIREMENTS **A**NALYSIS AND **S**PECIFICATION **D**OCUMENT



Most relevant domain assumptions

DA3 The user who reserves the car will always be the person who drives it

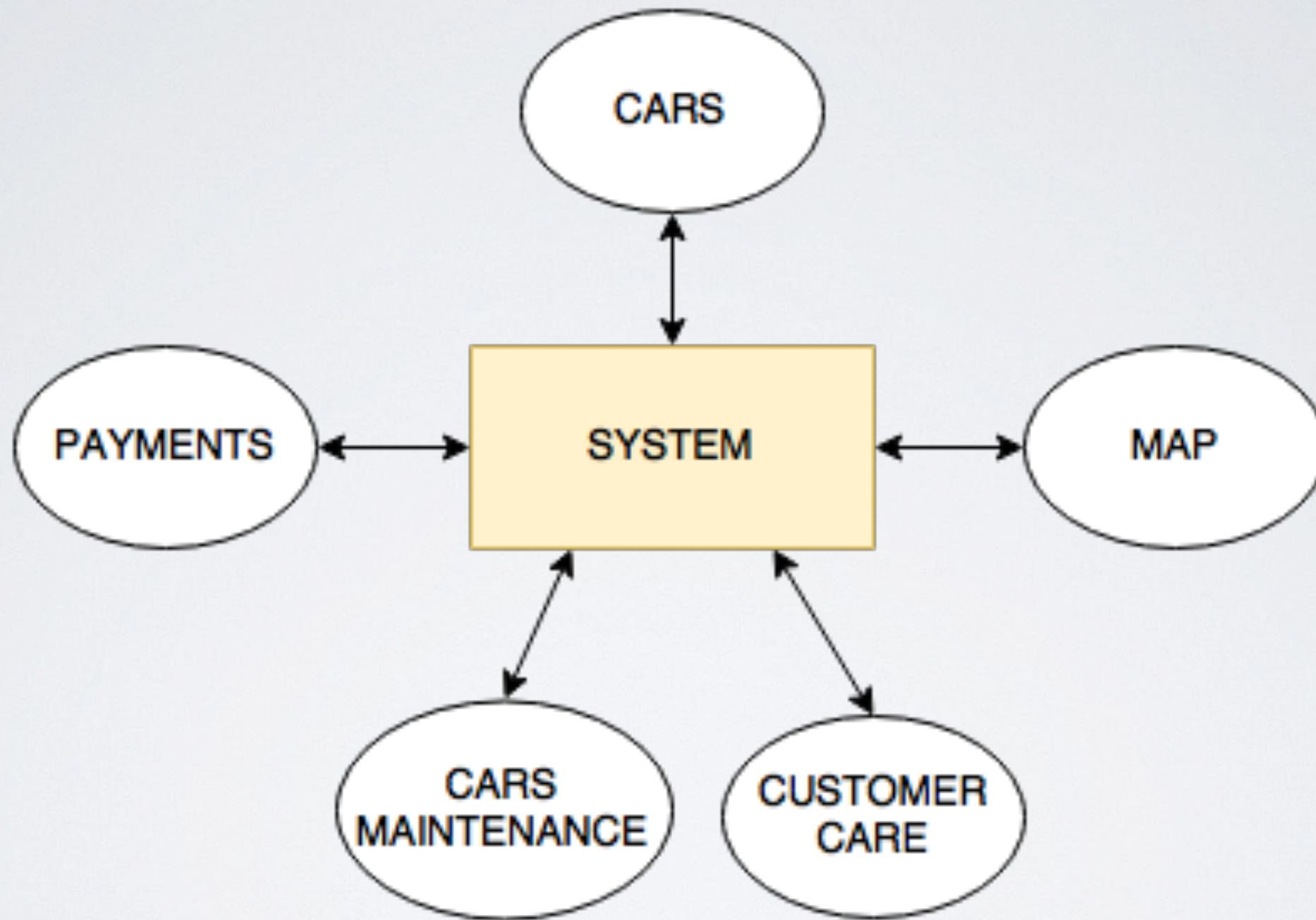
DA4 Users are legally allowed to drive cars (i.d. users have a proper driving license)

DA6 All data provided by users are correct and reliable

DA8 Charging station can exclusively be used by PowerEnjoy cars

DA11 All charging stations are located inside a safe area

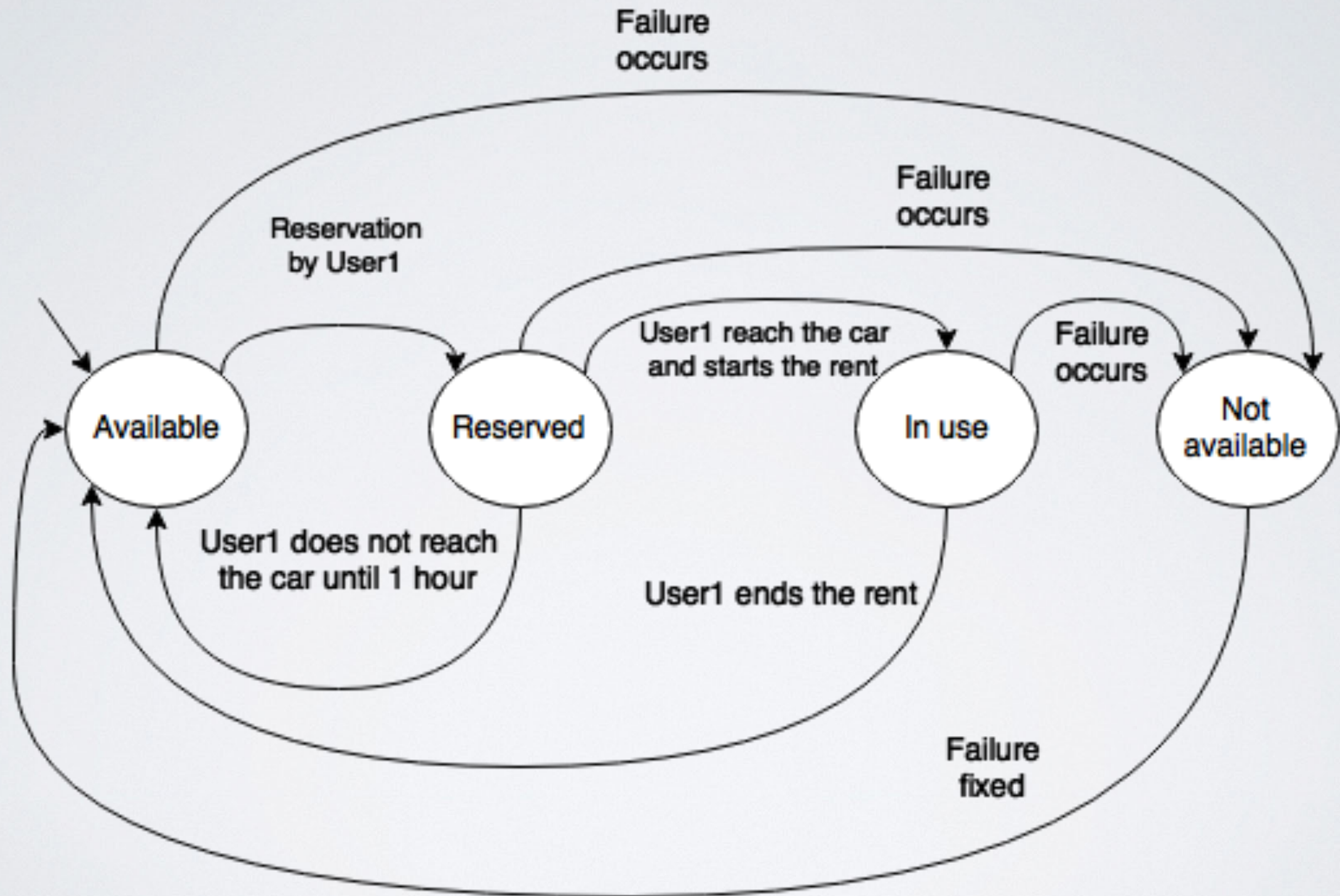
System Interfaces



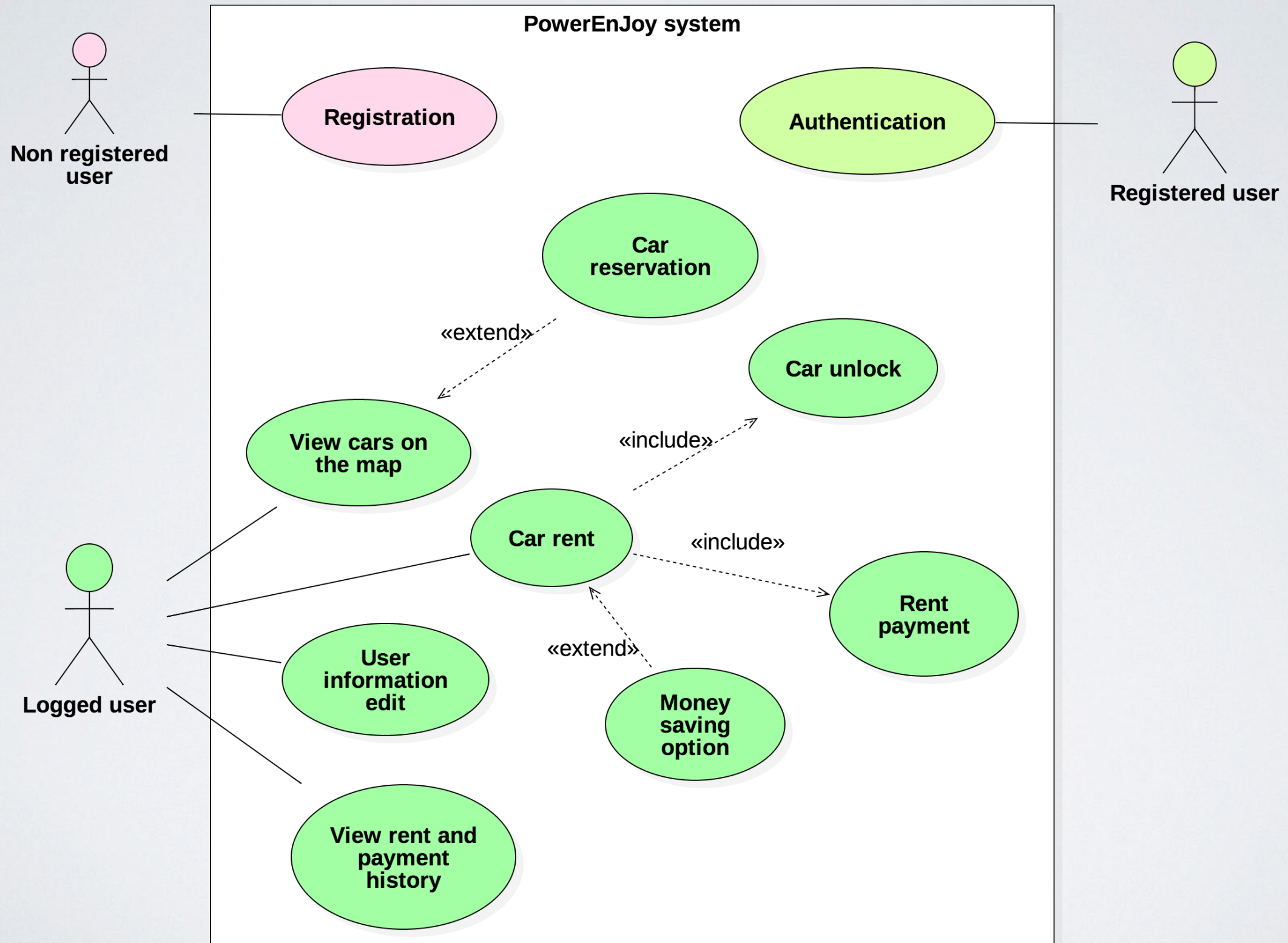
Cars

- Provided by the customer
- Equipped with a communication system
- Equipped with a **module** which provides a **set of primitives** runnable through a dedicated API
- A **built-in user interface** embedded with the modules shows some relevant information to the user

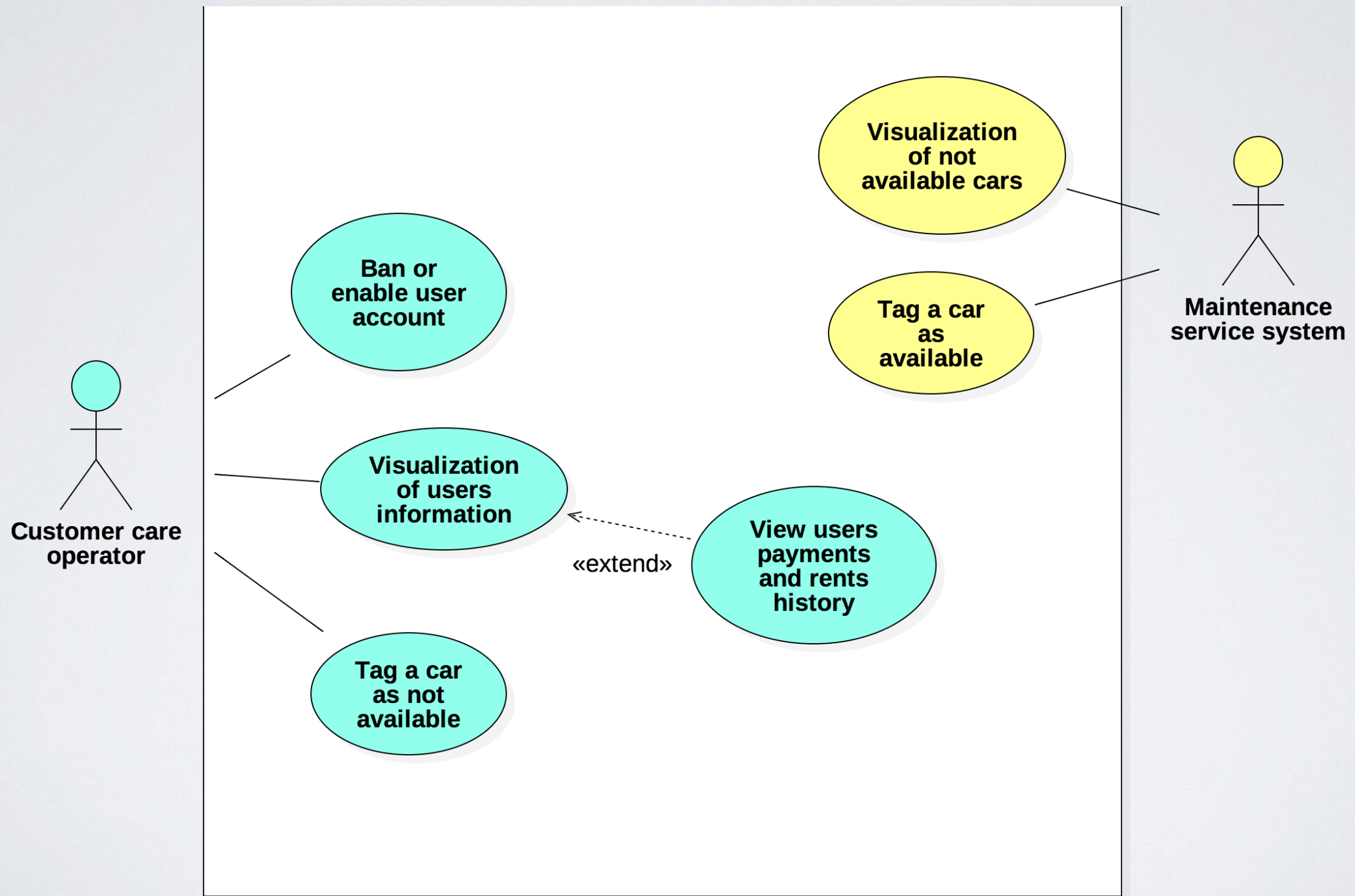
Car FSM



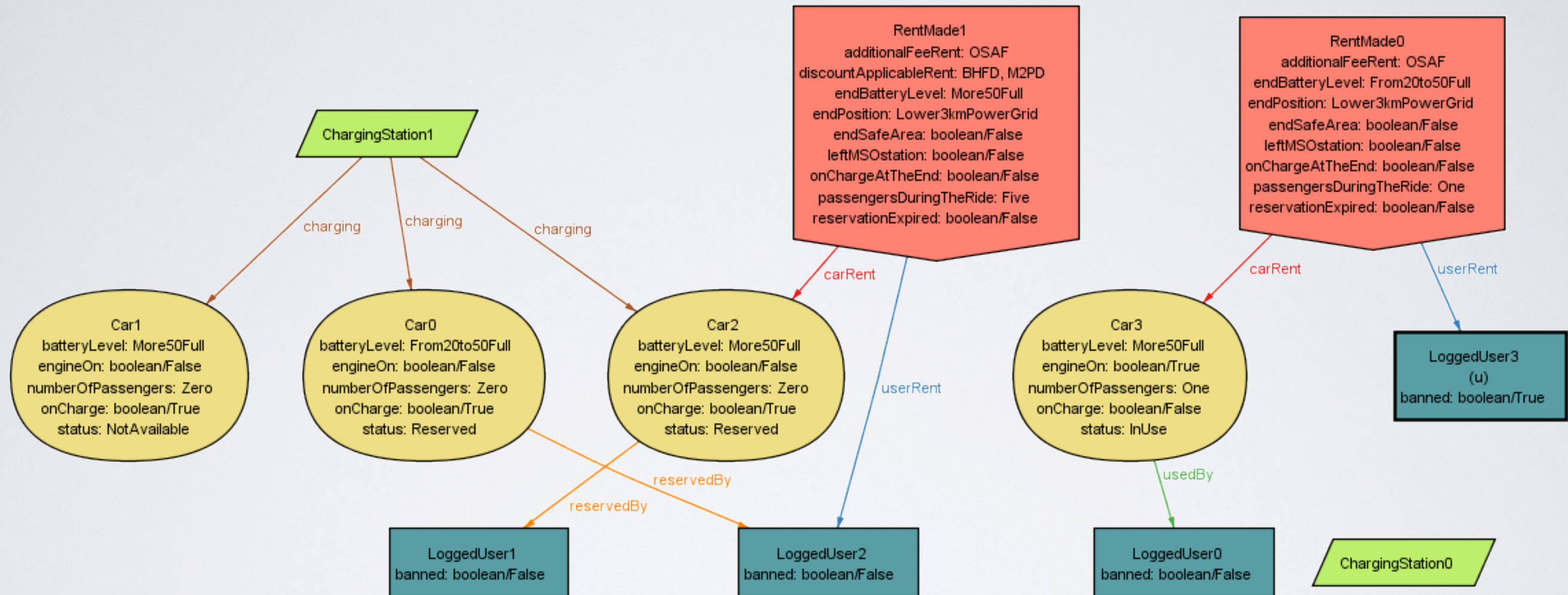
Use Cases



Use Cases

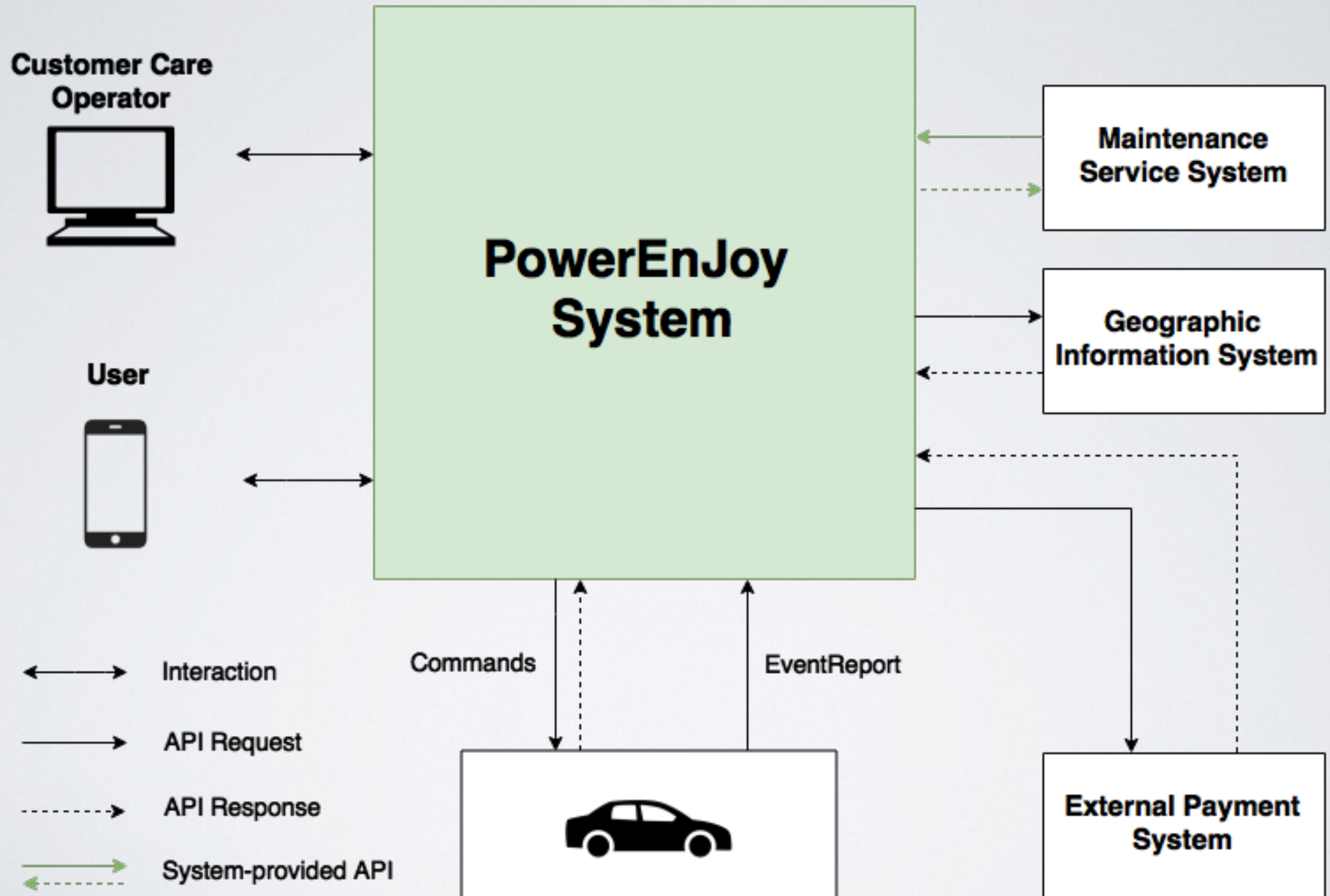


An Alloy world example

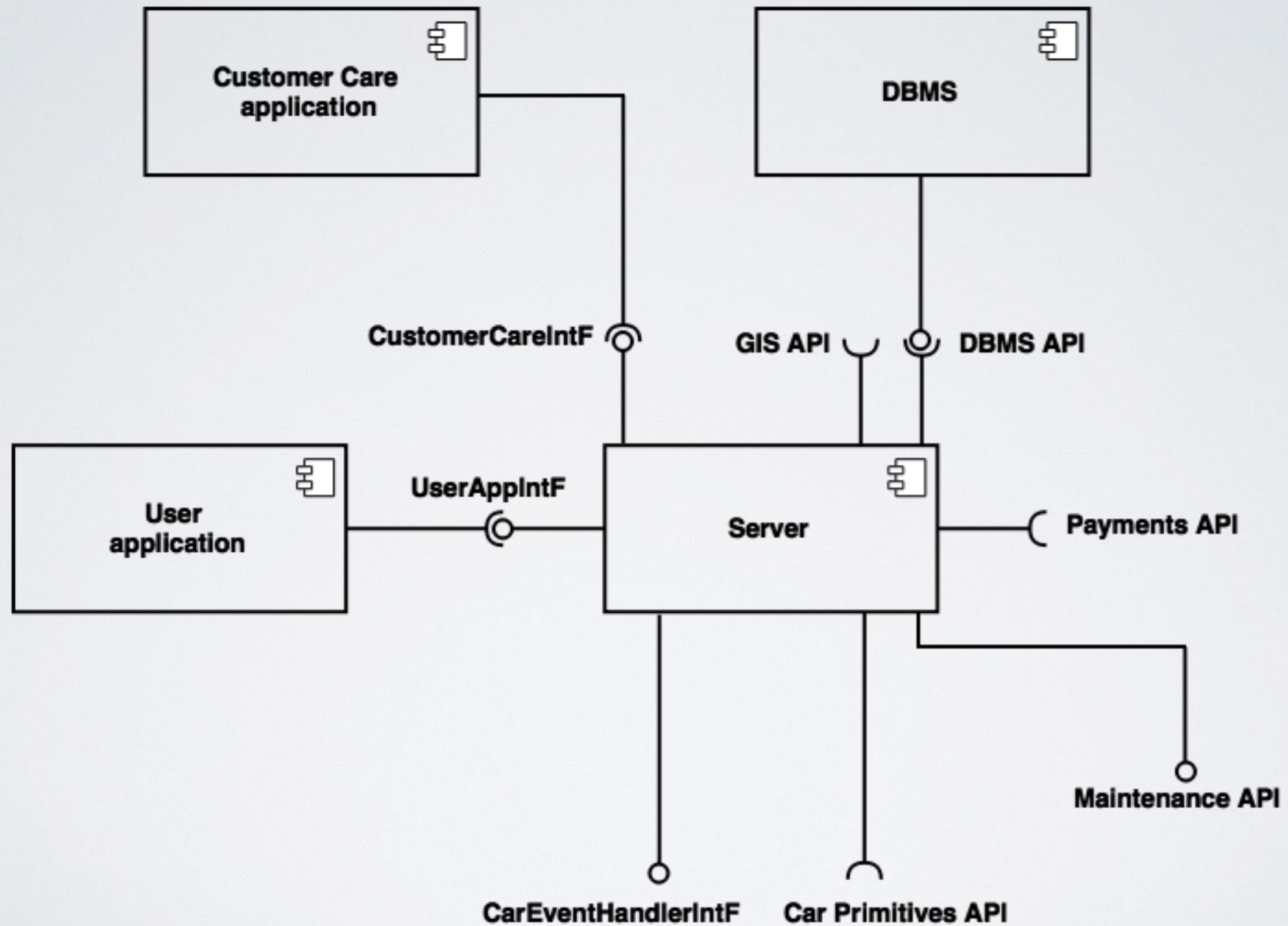


DESIGN DOCUMENT

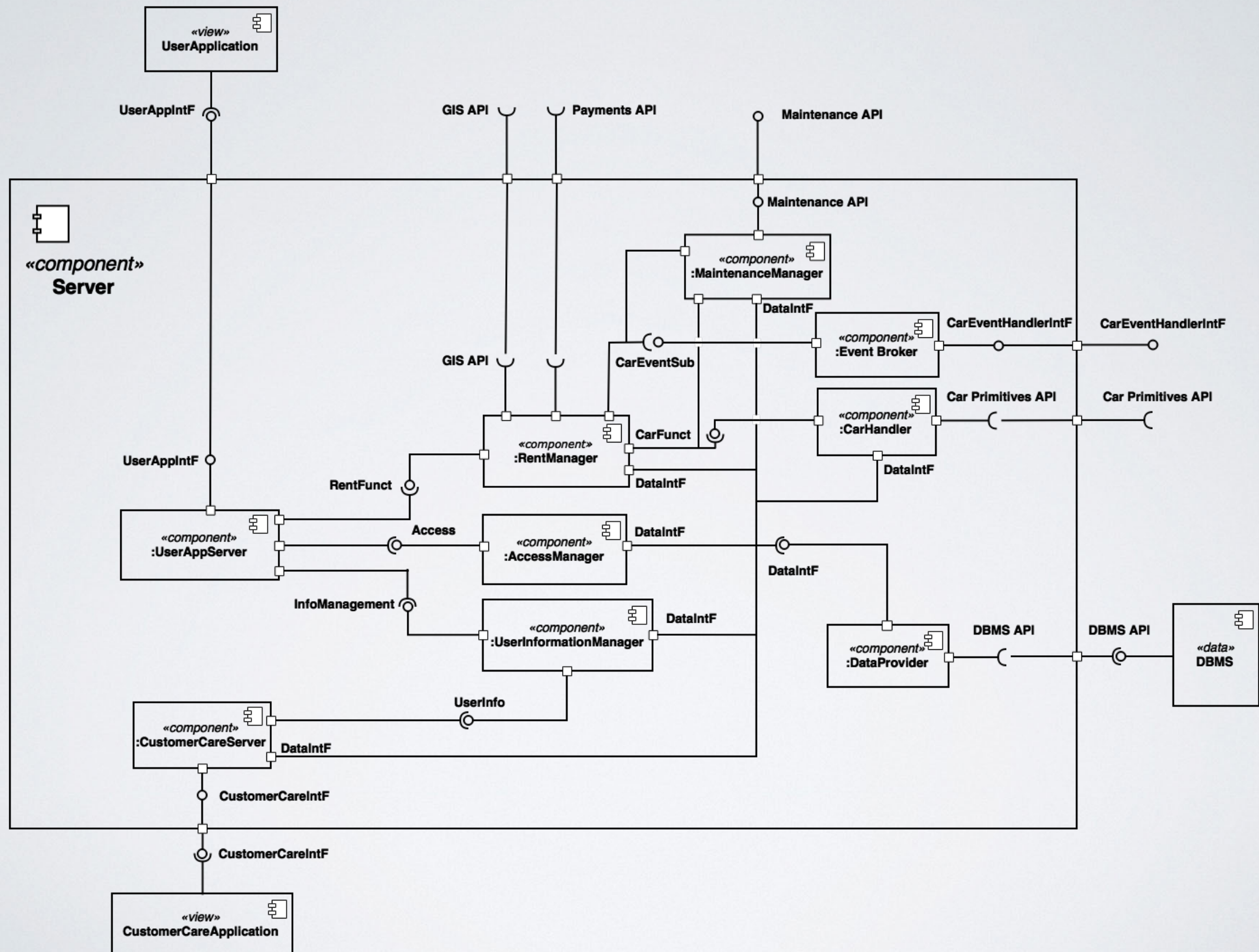
High-level context view point



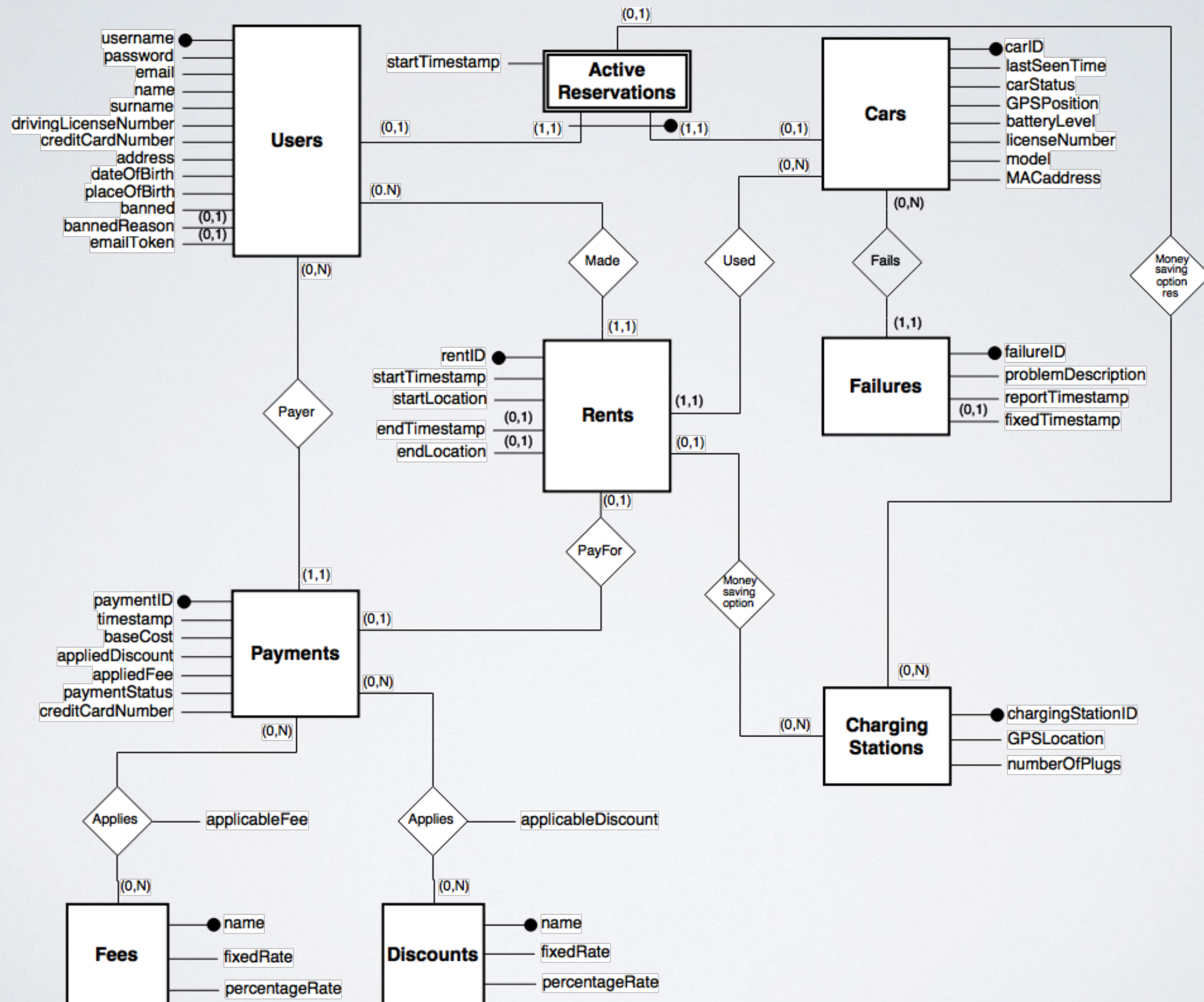
Main component



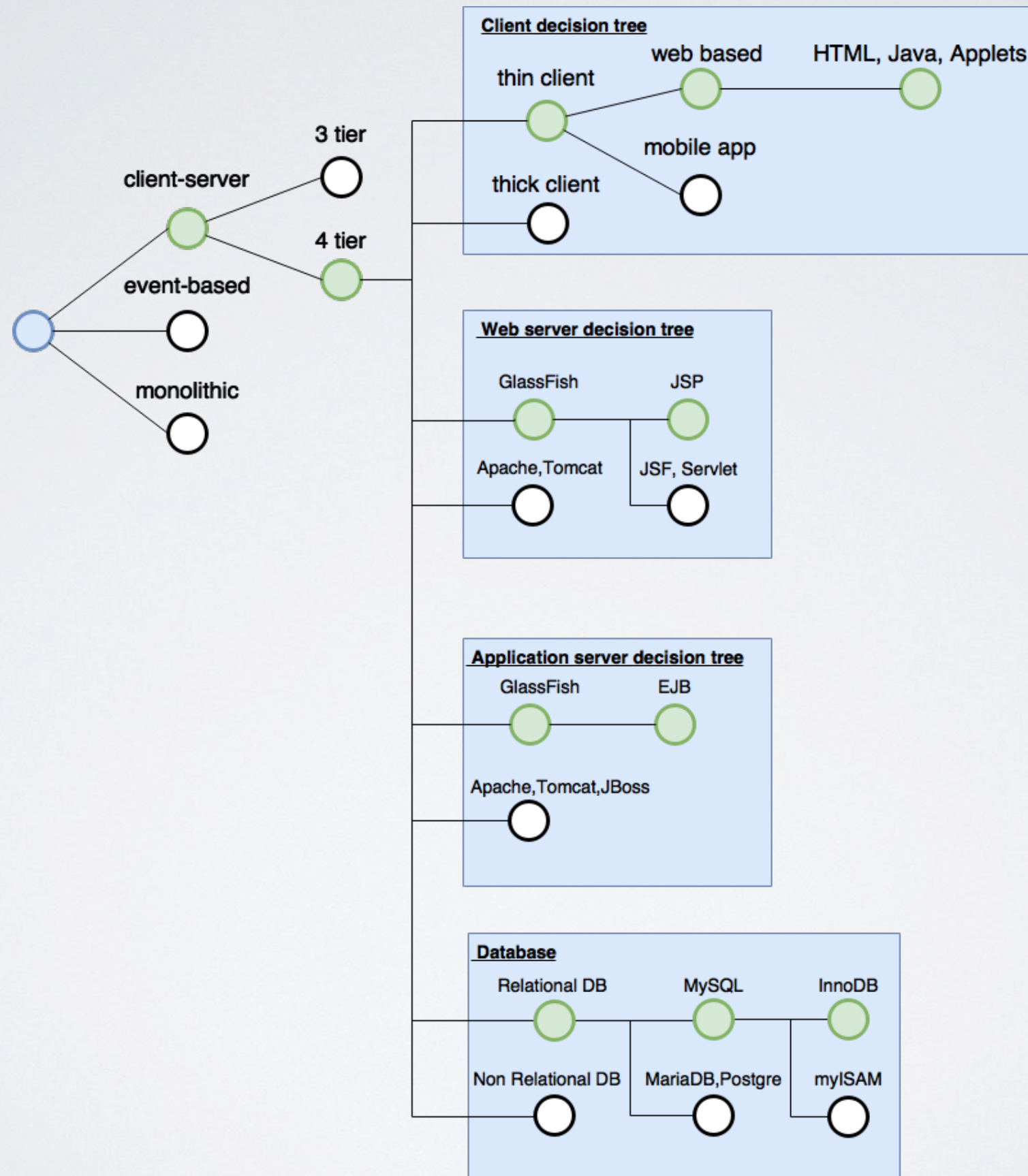
Server component *component diagram*



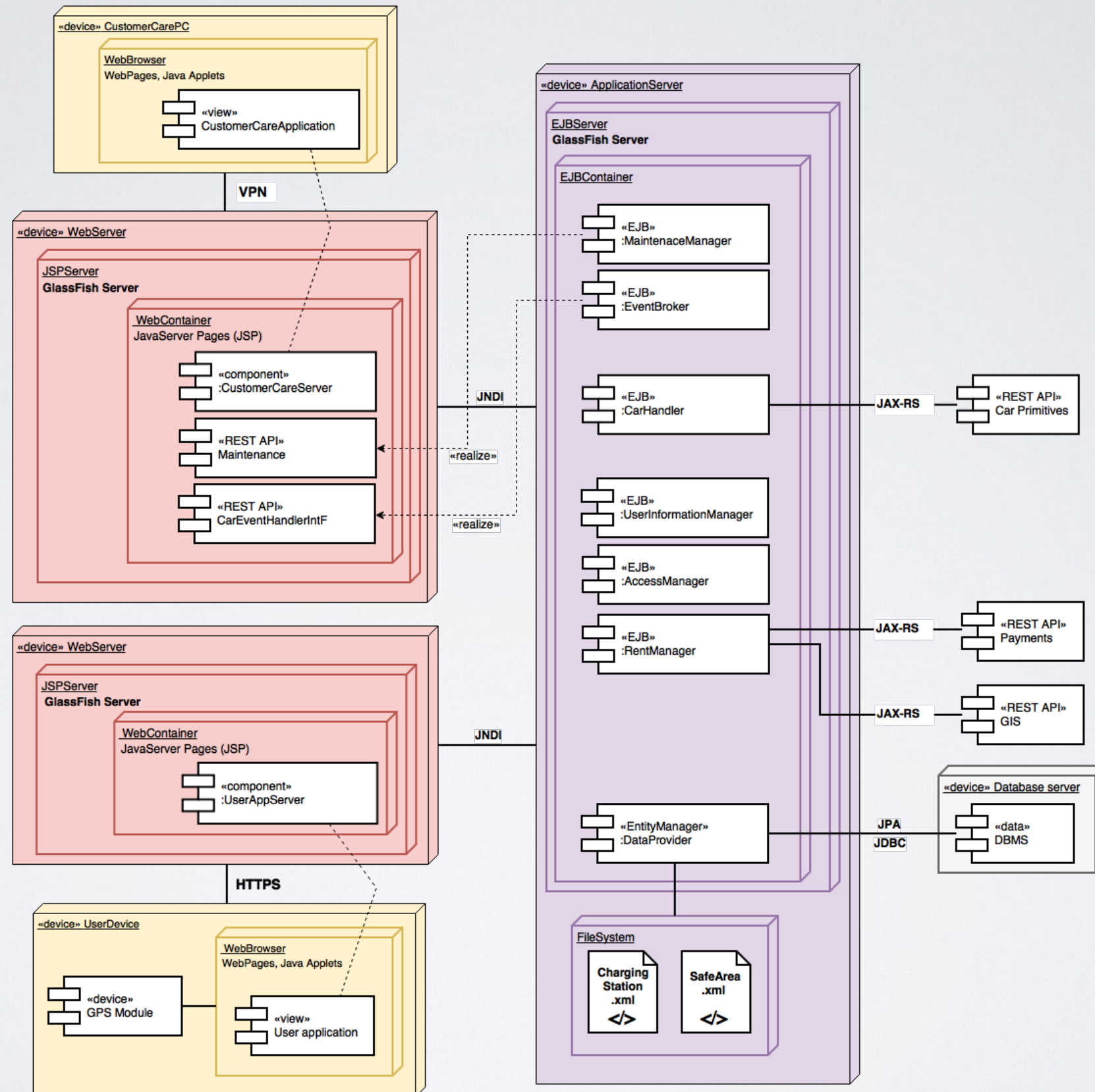
ER diagram



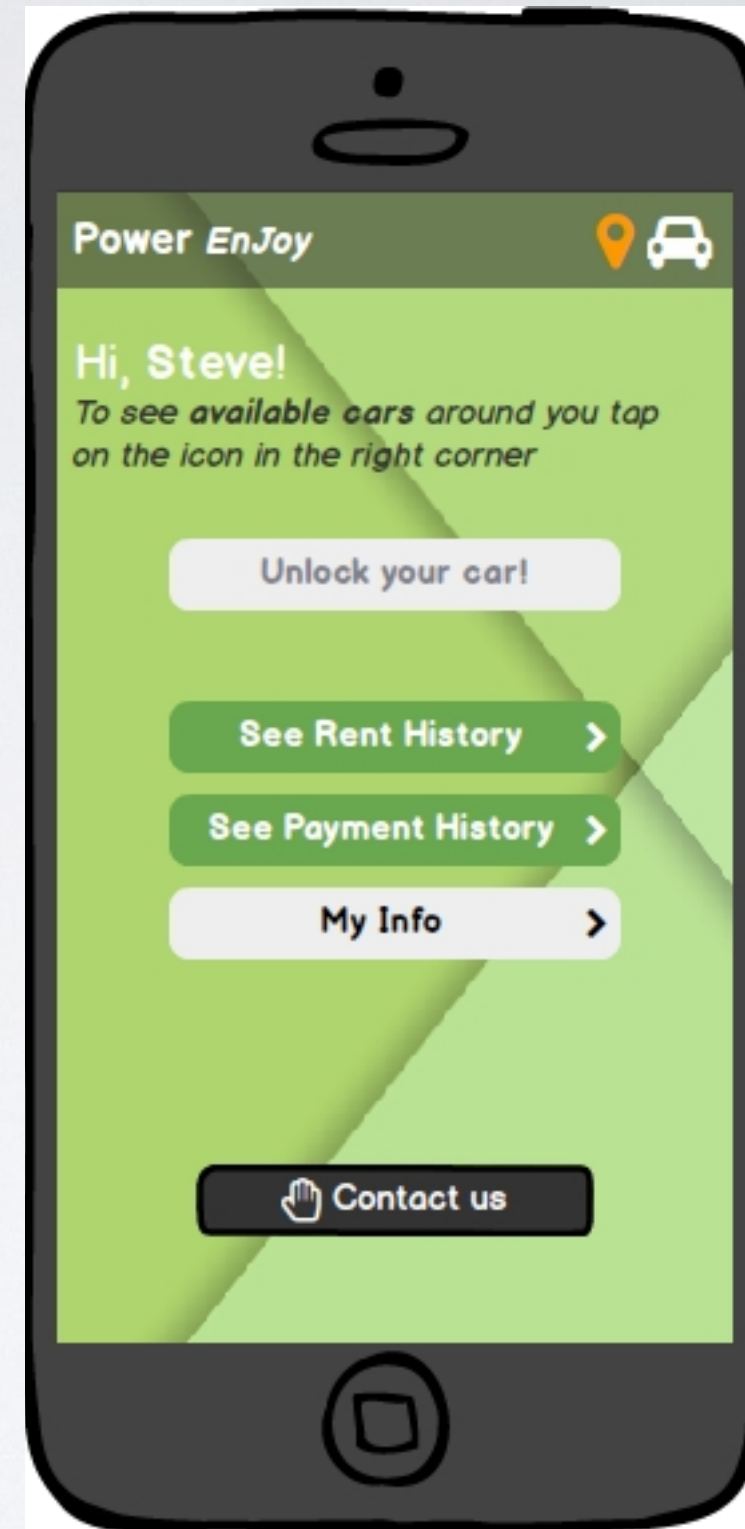
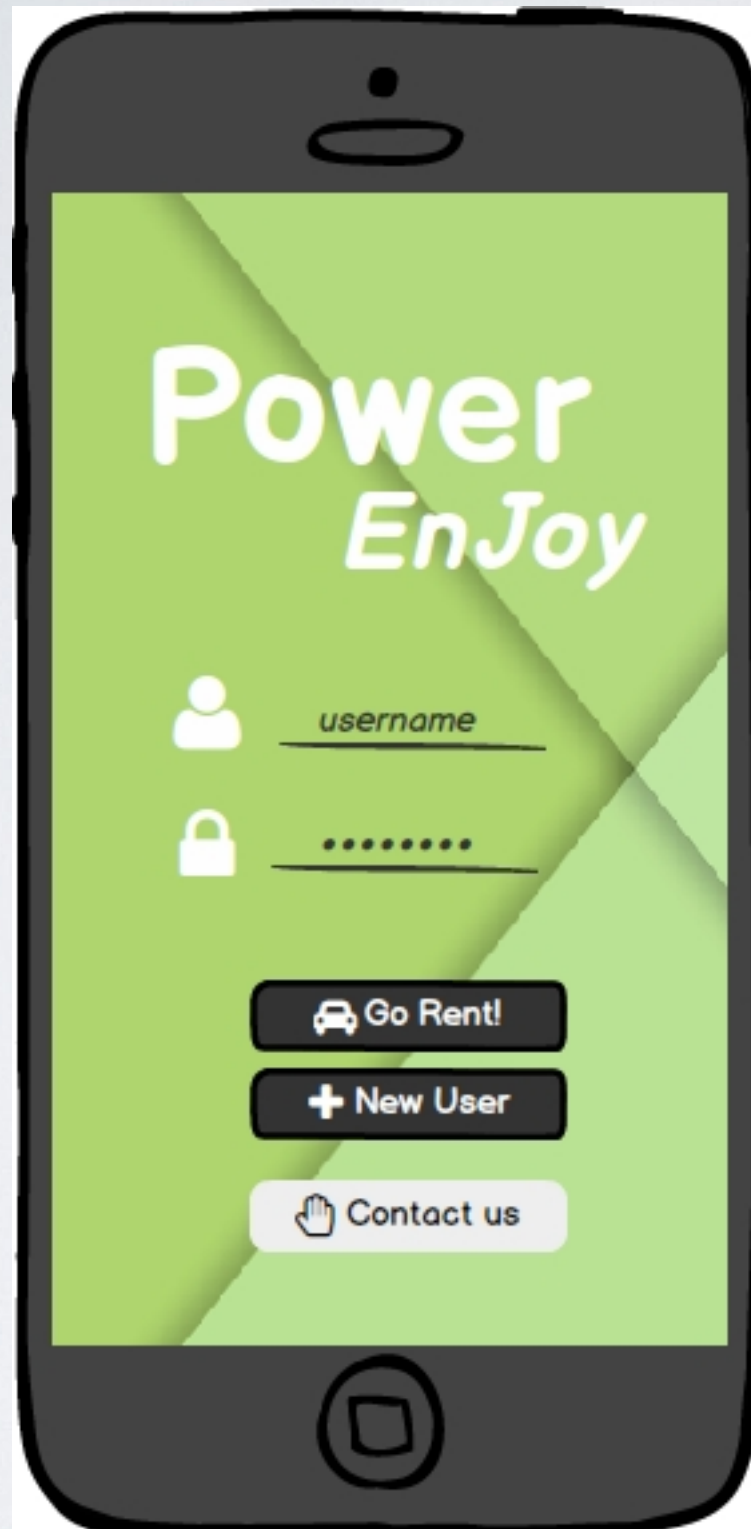
Decision Tree



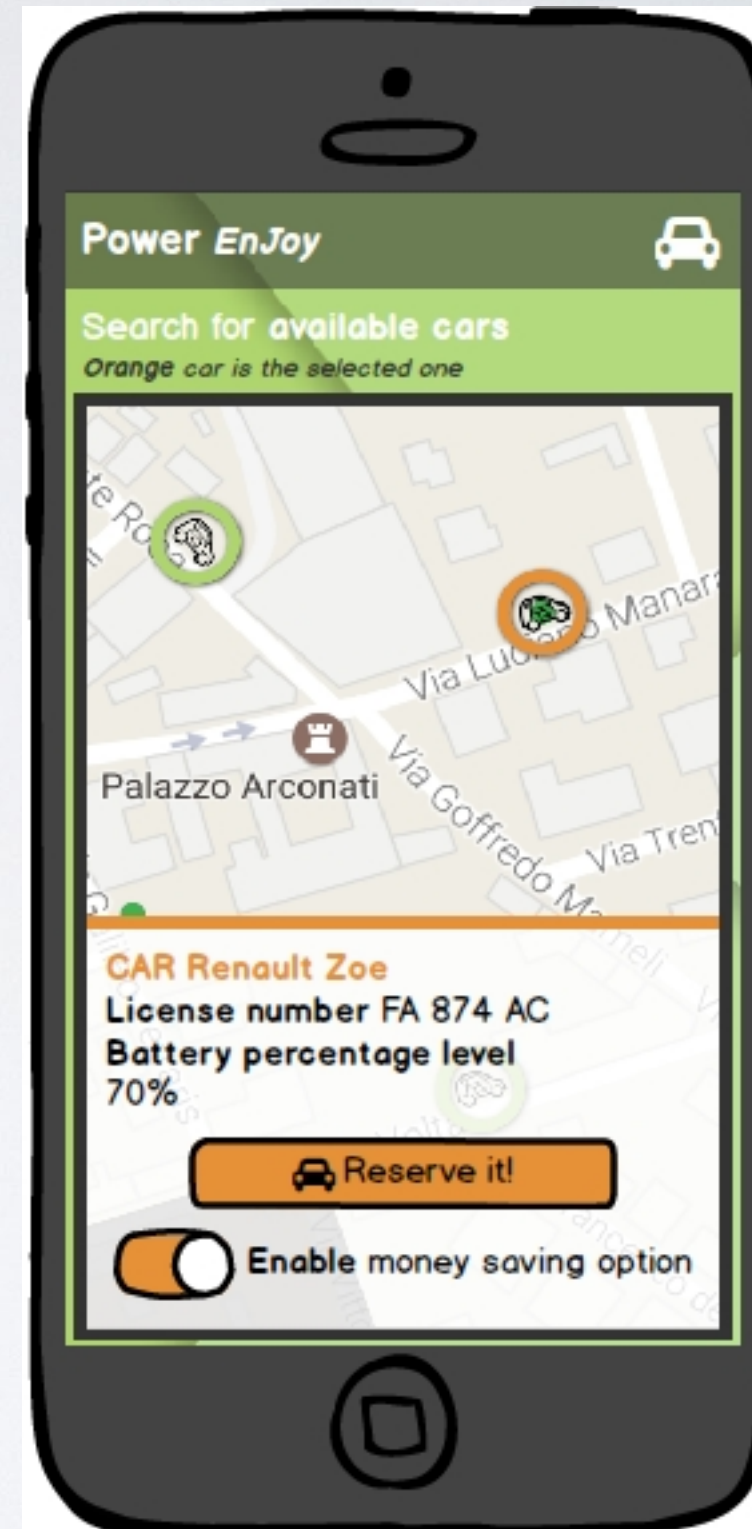
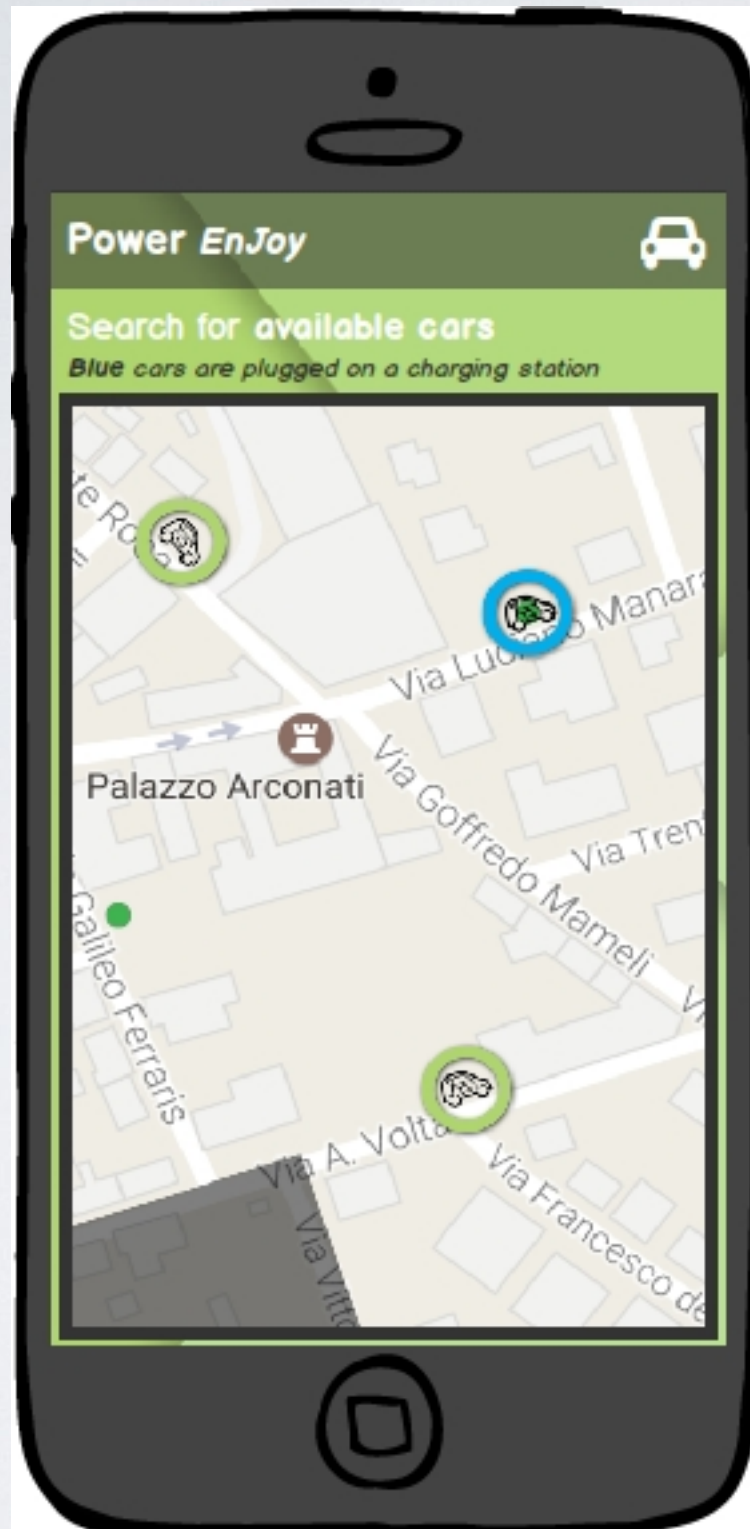
Deployment diagram



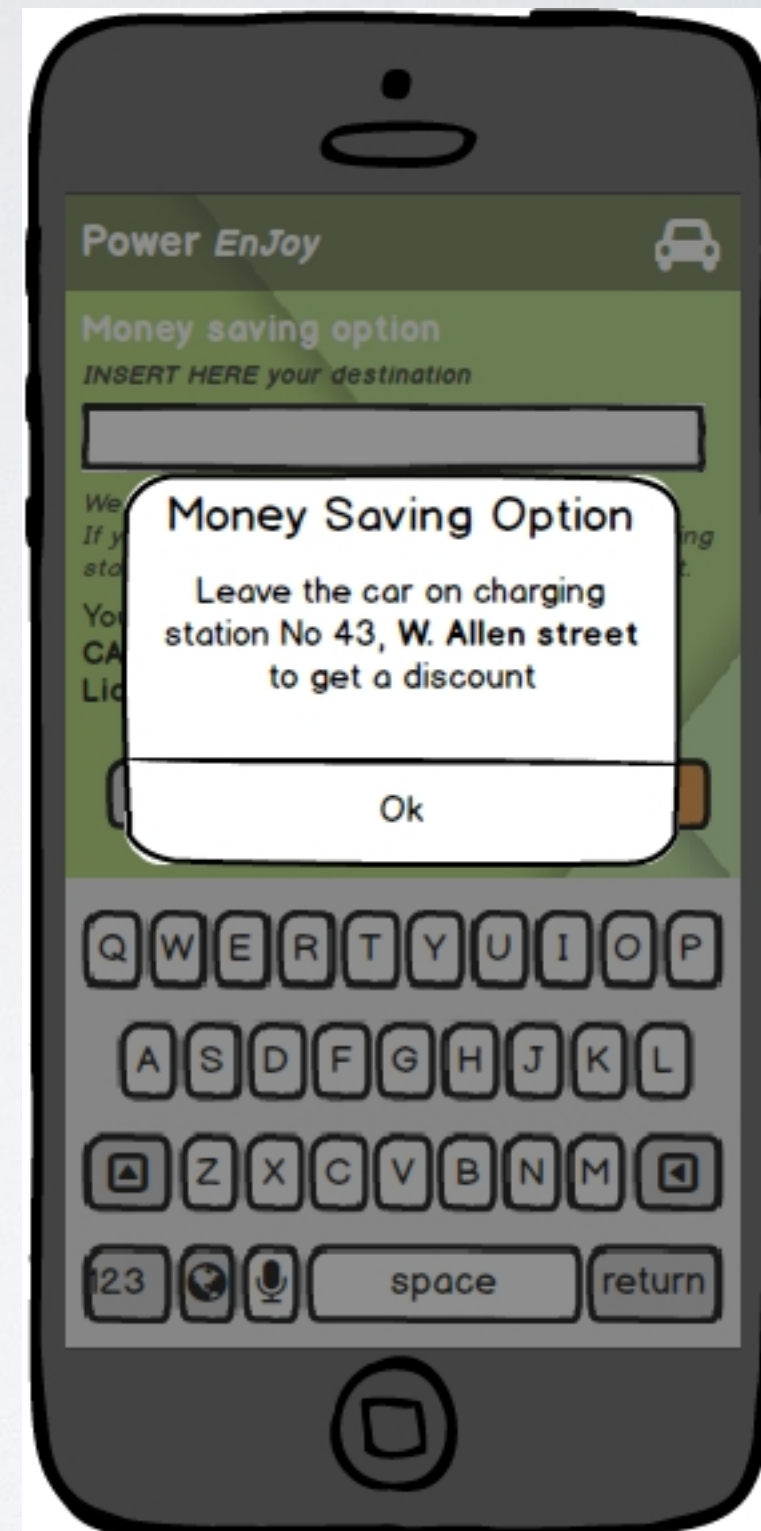
Login and Home mockups



Reserve a car mockups



Money saving option mockups



Customer care home mockup

Customer Care | Operator: 567656

←

→

✕

🏠

http://172.18.0.1

🔍

Power EnJoy

Customer Care App

🚗

Users

View user's info

Insert username or email

➡

Mark/Unmark user as banned

Insert username

➡

👤 Obtain all users

Cars

Tag car as Not Available

Insert license number

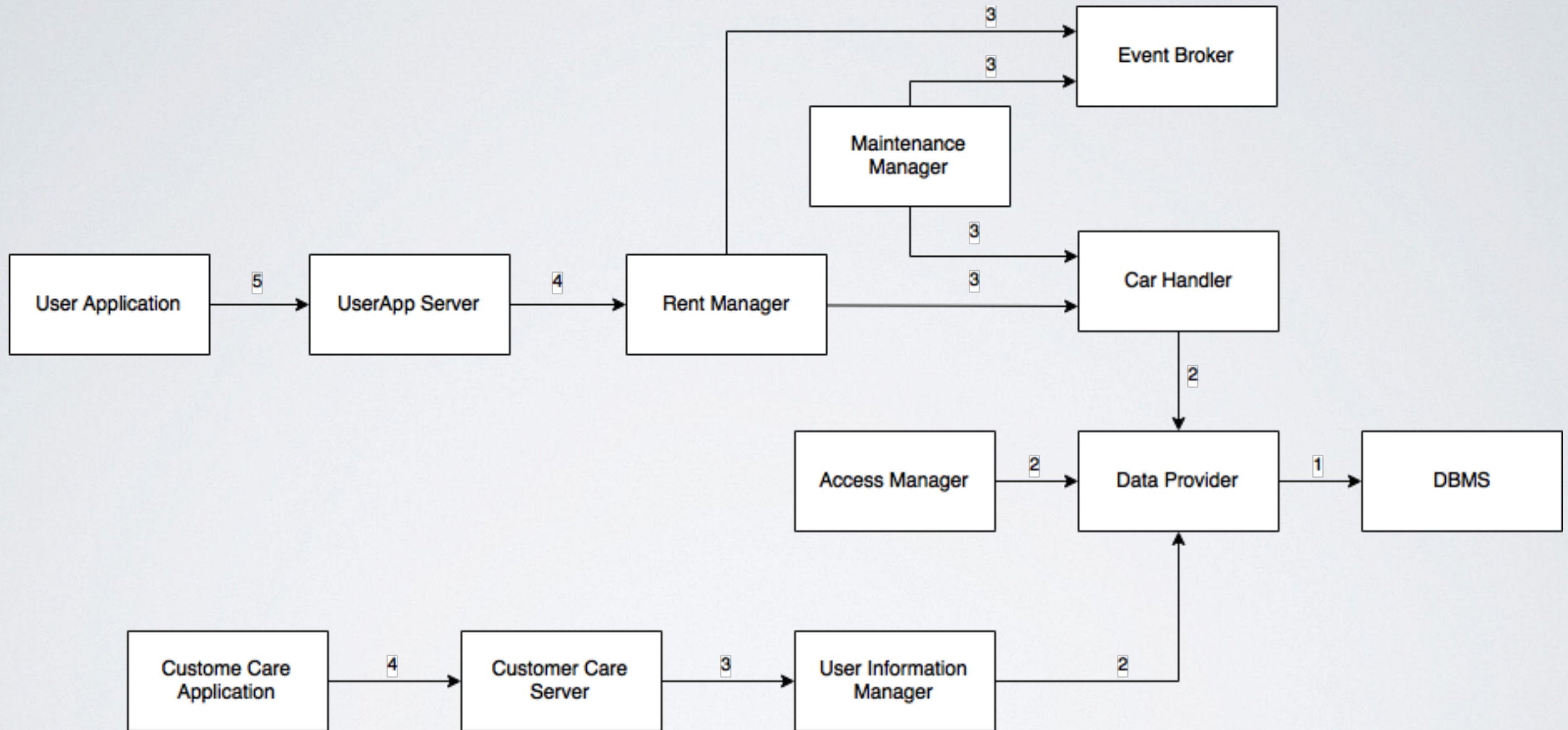
➡

INTTEGRATION **T**ESTING **P**LAN **D**OCUMENT

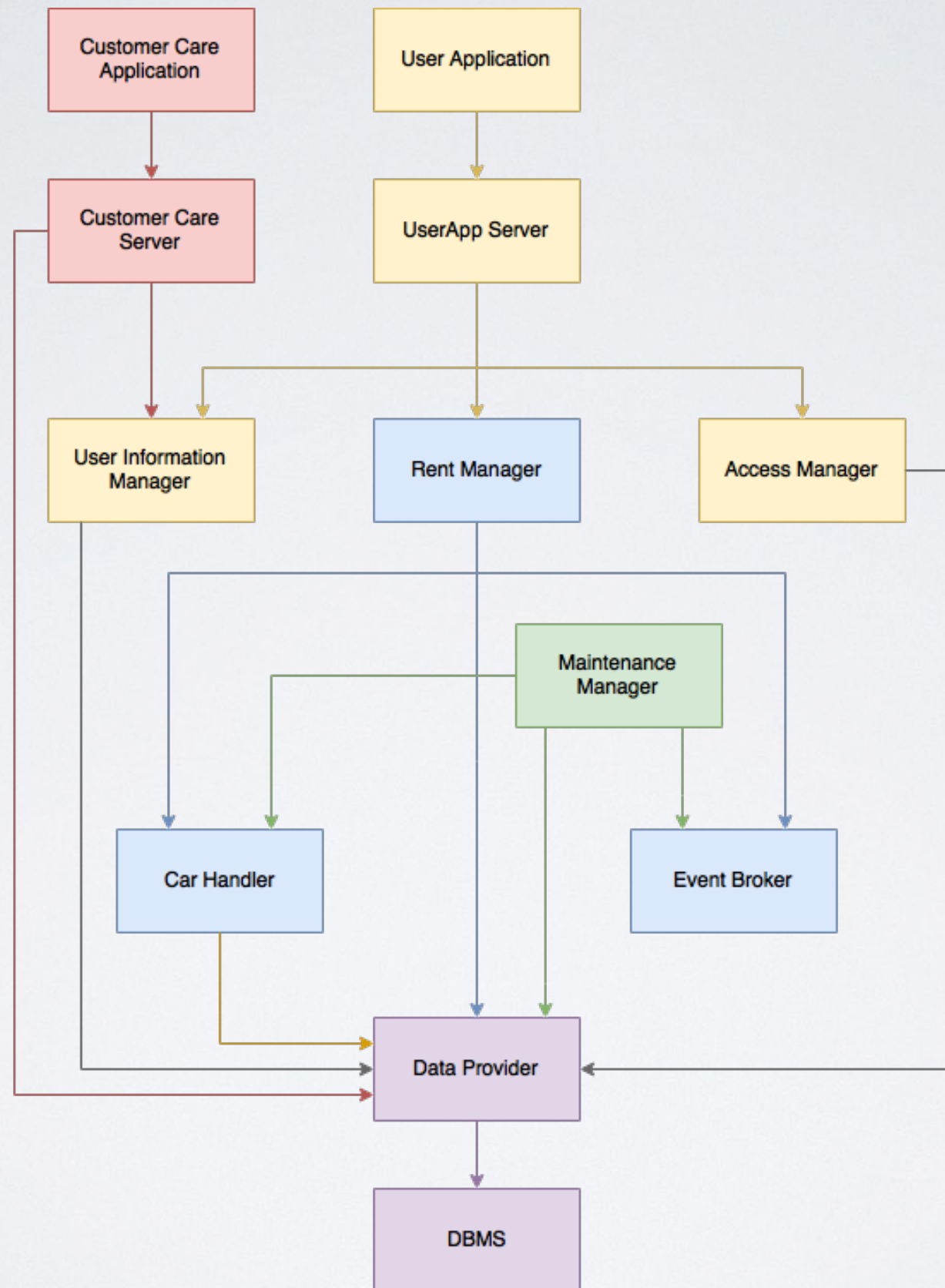
Integration testing strategy

- Most complex and relevant functionalities are located in the lower end tiers, which are also the most independent ones
- **Bottom-up approach**
- Critical-module-first approach in case of same priority
- Given our choices the most independent components are the only ones that needs to be fully developed at the time of starting the integration process
- Necessity of Drivers

Precedences diagram



Integration sequence



Integration tools

- **Unit Testing:** **JUnit** for the testing framework in combination with **Mockito** as a mocking framework
- **Integration Testing:** JUnit and Mockito, but in this case in combination with the **Arquillian** integration testing framework
- **Performance Testing:** load testing tool JMeter

PROJECT **P**LANNNING

Estimations

In the Project Plan document we gave an estimation for the size, cost and effort of the project based upon techniques that support the estimation procedure:

- The **Function Points** method is used to estimate the size of the project
- The **COCOMO II** method is used for cost and effort estimations

We report here the results of the estimation activities

Function Points

The following table summarizes FPs count identified through the analysis for each component.

Function Type Value	FPS
Internal Logic Files	49
External Interface Files	14
External Inputs	56
External Outputs	21
External Inquiries	28
Total	168

Function Points

Through the *SLOC* estimation formula we can therefore estimate an average and an upper bound for the size of the project.

$$SLOC = \left(\sum_{i \in I} \sum_{j \in i} w_{ij} \right) \cdot K$$

where

- $I = \{ILF, EIF, EI, EO, EQ\}$
- w_{ij} is the weight associated with the j -th function component of type i
- K is the industry gearing distribution of J2EE $\frac{SLOC}{FP}$ factor

$$SLOC_{avg} = \left(\sum_{i \in I} \sum_{j \in i} w_{ij} \right) \cdot K = 168 \cdot 46 = 7728$$

$$SLOC_{max} = \left(\sum_{i \in I} \sum_{j \in i} w_{ij} \right) \cdot K = 168 \cdot 67 = 11256$$

COCOMO II

The COCOMO II effort equation can now be computed as

$$\begin{aligned} PM &= A \cdot Size^E \cdot \prod_{i \in CostDrivers} C_i \\ &= 2.94 \cdot Size^{1.0761} \cdot (1 \cdot 1.33 \cdot 1 \cdot 1 \cdot 1.22 \cdot 0.87 \cdot 1) \\ &= 2.94 \cdot Size^{1.0761} \cdot 1.4112 \end{aligned}$$

Therefore

$$PM_{avg} = 2.94 \cdot 7.728^{1.0761} \cdot 1.4112 = 37$$

$$PM_{max} = 2.94 \cdot 11.256^{1.0761} \cdot 1.4112 = 56$$

COCOMO II

According to the model definition, the duration D (in months) of the project can be estimated as

$$D = 3.67 \cdot (PM_{NS})^F \cdot \frac{SCED\%}{100}$$

where

- PM_{NS} is the Person-Months estimation without considering the SCED cost driver
- $F = 0.28 + 0.2 \cdot (E - 0.91)$
- E is the aggregation of scale factors previously determined
- $SCED\%$ is the SCED nominal value percentage, in our case is set to 100%

Since the effort multiplier of SCED cost driver is 1, it results

$$PM_{NS} = PM$$

Since as previously specified $E = 1.0761$ we can compute

$$F = 0.28 + 0.2 \cdot (1.0761 - 0.91) = 0.31322$$

Therefore

















$$D_{avg} = 3.67 \cdot 37^{0.31322} \cdot 1 = 11.373 \text{ months}$$

$$D_{max} = 3.67 \cdot 56^{0.31322} \cdot 1 = 12.949 \text{ months}$$

RASD GANTT

ID	Task Name	Start	Finish	Duration	ott 2016			nov 2016	
					16/10	23/10	30/10	6/11	
1	RASD	17/10/2016	11/11/2016	20d					
2	Meetings with client	17/10/2016	19/10/2016	3d					
3	Identification of goals and stakeholders	17/10/2016	17/10/2016	1d					
4	Requirements elicitation	18/10/2016	19/10/2016	2d					
5	Modelization of the World and the Machine	20/10/2016	25/10/2016	4d					
6	Identification of domain assumptions	20/10/2016	25/10/2016	4d					
7	Identification of system goals	20/10/2016	25/10/2016	4d					
8	Identification of requirements	20/10/2016	25/10/2016	4d					
9	Writing scenarios	26/10/2016	26/10/2016	1d					
10	Alloy model draft	26/10/2016	28/10/2016	3d					
11	Identification of needed data	27/10/2016	27/10/2016	1d					
12	Identification of use cases	27/10/2016	28/10/2016	2d					
13	In progress meeting with client	02/11/2016	02/11/2016	1d					
14	Document refinement	03/11/2016	07/11/2016	3d					
15	Requirements refinement	03/11/2016	07/11/2016	3d					
16	Data model refinement	03/11/2016	04/11/2016	2d					
17	Sequence diagrams for use cases	03/11/2016	07/11/2016	3d					
18	Alloy modelization	03/11/2016	07/11/2016	3d					
19	Document revision	08/11/2016	11/11/2016	4d					

DD GANTT

ID	Task Name	Start	Finish	Duration	nov 2016			dic 2016
					13/11	20/11	27/11	
1	DD	14/11/2016	02/12/2016	15d				
2	Architecture draft	14/11/2016	16/11/2016	3d				
3	High level system view definition	14/11/2016	16/11/2016	3d				
4	Definition of system boundaries	14/11/2016	16/11/2016	3d				
5	Main architecture decisions	14/11/2016	16/11/2016	3d				
6	Meeting with clients	17/11/2016	17/11/2016	1d				
7	Refining architecture choices	18/11/2016	21/11/2016	2d				
8	Main components component diagram	22/11/2016	23/11/2016	2d				
9	ER DB Model	24/11/2016	24/11/2016	1d				
10	Components internal structure	24/11/2016	25/11/2016	2d				
11	Components external interfaces	24/11/2016	25/11/2016	2d				
12	Mapping components to goals	28/11/2016	28/11/2016	1d				
13	Sequence diagrams	25/11/2016	30/11/2016	4d				
14	Mockups	29/11/2016	30/11/2016	2d				
15	Deployment diagram	01/12/2016	01/12/2016	1d				
16	Algorithms desing	01/12/2016	01/12/2016	1d				
17	Revision and requirement traceability	02/12/2016	02/12/2016	1d				

ITPD GANTT

ID	Task Name	Start	Finish	Duration	dic 2016	
					11/12	18/12
1	ITPD	12/12/2016	19/12/2016	6d		
2	Integration test strategy	12/12/2016	12/12/2016	1d		
3	Definition of precedences	12/12/2016	12/12/2016	1d		
4	Sequence of components integration	13/12/2016	14/12/2016	2d		
5	Program Drivers	15/12/2016	15/12/2016	1d		
6	Tools and equipment	15/12/2016	15/12/2016	1d		
7	Integration test description	13/12/2016	15/12/2016	3d		
8	Data required	16/12/2016	16/12/2016	1d		
9	Document revision	16/12/2016	19/12/2016	2d		

Risks

Risk (probability, effects)

Project risks

- Failure to achieve a deadline (Moderate, Moderate)
- Personnel shortfall (Low, Serious)
- Change of requirements (Moderate, Moderate/Serious)

Business risks

- Competitors (High, Serious)
- Financial crisis (Low, Catastrophic)

Risks

Technical risks

- Underestimation of requests' load (Low, Serious)
- Data Loss (Low, Catastrophic)
- Wrong external component specification (Low, Serious)
- Difficulties during development phase (Low, Moderate)
- Car supplier changes (Low, Serious)
- External APIs bad behaviour (Low, Moderate)
- External APIs downtime (Low/Moderate, Serious)

CODE ■ INSPECTION

Apache OfBiz project

The OFBiz project is an **open source** project by Apache that includes a suite of enterprise applications such as a **ERP** (Enterprise Resource Planning), **CRM** (Customer Relationship Management), E-Business / E-Commerce, SCM (Supply Chain Management), MRP (Manufacturing Resource Planning), MMS/EAM (Maintenance Management System/Enterprise Asset Management).

These applications are built on a **common architecture** using common data, logic and process components. The **loosely coupled nature** of the applications makes these components easy to understand, extend and customize [1].

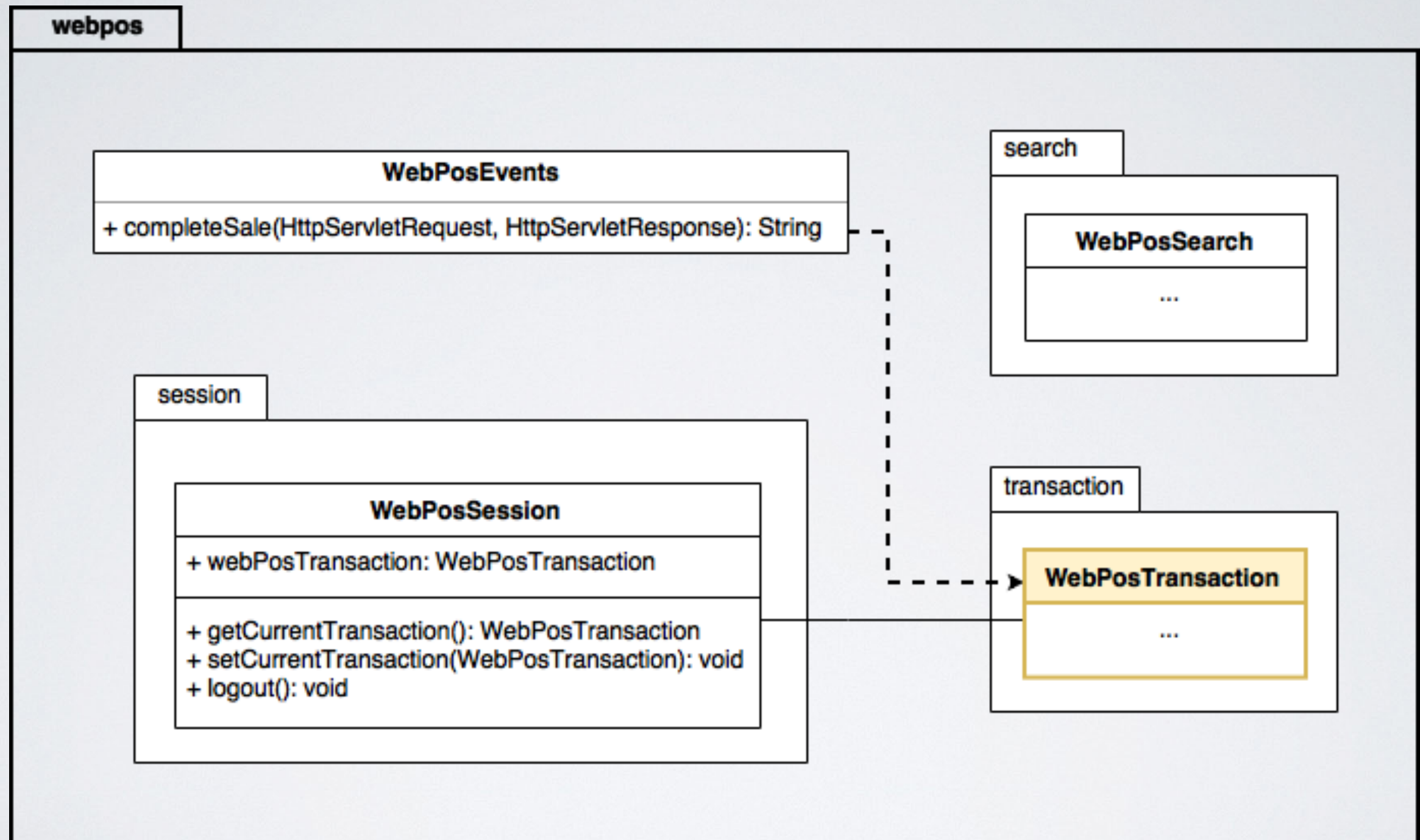
[1] Taken from <https://ofbiz.apache.org/apache-ofbiz-project-overview.html>

The *webpos* package

This package is related to a plugin that may be integrated with the Apache OFBiz suite of applications.

- **org.apache.ofbiz.webpos.search** package
Contains a java class to manage some searching functionalities (e.g. about facilities and products)
- **org.apache.ofbiz.webpos.session** package
Contains a java class representing a web pos session.
- **org.apache.ofbiz.webpos.transaction** package
Contains a java class representing a single web pos transaction
- **org.apache.ofbiz.webpos.WebPosEvents.java** class
Java class to manage incoming requests from the related web based application

The *WebPosTransaction.java* class



Main issues found

Here we report the major issues we found, while performing the code inspection:

- Complete absence of javadoc or comments
- Incomplete methods and catch blocks
- Presence of unclear variable names
- Almost complete absence of high order brakes, and high number of long lines (above 80 chars)

Checklist related issues

Initialization and Declarations

- *Check that all object references are initialized before use*

L 67: the session object passed as parameter in the constructor is referenced and used several times in the class but it has never checked not to be null. This might cause a NullPointerException.

```
65     private WebPosSession webPosSession = null;
66
67     public WebPosTransaction(WebPosSession session) {
68         this.webPosSession = session;
```

Output Format

- *Check that error messages are comprehensive and provide guidance as to how to correct the problem*

L 396: in the processExternalPayment method no error message is returned, marked as todo.

```
396         if (refNum == null) {
397             //TODO handle error message
398             return;
399         }
```


Checklist related issues

Computation, Comparisons and Assignments

- *Check that the implementation avoids "brutish programming"*

L 53, L 54, L 55: it is better to define an enumeration than use three public static final int variables to define the type of payment

```
53      public static final int NO_PAYMENT = 0;  
54      public static final int INTERNAL_PAYMENT = 1;  
55      public static final int EXTERNAL_PAYMENT = 2;
```

Checklist related issues

Exceptions

- *Check that the appropriate action are taken for each catch block*
- All catch blocks only log the exception to the debugger. Moreover
- L 390:** in the processNoPayment method the catch (GeneralException e) block is empty

```
390      } catch (GeneralException e) {  
391          // errors handled  
392      }
```

- L 407:** in the processExternalPayment method the catch (GeneralException e) block is empty

```
407      } catch (GeneralException e) {  
408          // errors handled  
409      }
```


Other issues found

- **COMPLETE TODO TASKS**

- **L 61**: the partyId field is private, it is initialized in the constructor of the class to the value “_NA_”, it is never updated and it does not offer any setter
- **L63**: the drawerIdx field is not used in the current implementation of the class, the related getter always returns one
- **L 207, L 214, L 224, L 316**: the usage of the wildcard <? extends Object> should be avoided
- **L 421, L 431**: the usage of type Object should be avoided
- **L 375, L 402**: it is better to create a new variable instead of reusing a method's parameter
- **L 289, L 436, L440**: getStoreOrgAddress and makeCreditCardVo methods return *null* values when an exception is raised during computation.

Thanks
for your attention