Software engineering 2 project

PowerEnJoy

Integration Test Plan Document (ITPD)

Authors:
Madaffari Federico (matriculation number: 792873)(Computer Science)
Mandrioli Claudio (matriculation number: 849973)(Automation and Control)

Table of Contents

Introduction

Revision history

This is the version 1.0 of this document.

Purpose and scope

This document is the Integration Testing Plan Document for the project PowerEnJoy. The integration testing should guarantee that each of the developed components works properly when combined with the others.
The purpose of this document is to present, starting from the proposed methodologies the procedure to be followed during the integration testing of the software product.

Document structure

The document is structured in five main sections:

- The 'Introduction' providing some information about the document itself
- The 'Integration strategy' that describes the proposed methodology for the testing. Firstly are provided the entry criteria for the testing phase, then starting from the structure proposed in the DD all the components taken into account during the structuring of the integration testing are listed. And in the last part of this section the sequence of integration tests is provided.
- The 'Individual steps an test description' lists and provides some description of all the test to be performed on the components.
- The 'Tools and test equipment required' and the 'Program stubs and test data required', as titles suggests, provide the list of software tools that will be required during the performing of the integration testing.

List of definition and abbreviations

In this section we provide some fundamental definitions for the readability of this document:

- API: application programming interface, it is a standard way to interact with another system.
- BEAN: is a server-side component that encapsulates a business logic of an application.
- JEE: Java Enterprise Edition, the computing platform for enterprise software used in this project.
- APPLICATION SERVER: software framework that provides facilities and environment to run mobile and web applications.
- MOBILE APPLICATION: the application supposed to run of devices like smartphones.

- WEB APPLICATION: the application supposed to run on computer devices.
- RASD: Requirements analysis and specification document, when named in this document we refer to the previously produced document for the PowerEnJoy service.
- DD: Design document previously produced for this project.
- SYSTEM: when this word is used in the document it refers to the digital management system to project
- USER: is the costumer of the PowerEnJoy service, not necessarily already registered to the system
- REGISTERED USER: is the user that registered to the system providing his personal info, therefore in the system there must be stored those data
- EMPLOYEES: are those people that work for the PowerEnJoy company that will take care of cars out of charge
- CAR: in this document my car are intended only those electric cars related to the PowerEnJoy service (model of car: BMW i3)
- START THE ENGINE: when the driver activates the electric motor of a car
- RIDE: is a time span in which a certain user is using a certain car. This time span starts when the user starts the engine (ride start) and ends when all doors of the car are closed and no passengers are on the car (ride end)
- PICK UP THE CAR: this expression is used as a synonym of 'start the ride'
- LOCK THE CAR: when the system (as only the system can perform this action) locks a car at the end of a ride
- UNLOCK THECAR: when the system (as only the system can perform this action) unlocks the car after the user's request
- RESERVE: when a user tells the system that he is going to use a car and therefore car must be marked as unavailable
- RESERVED CAR: a car marked unavailable because some user reserved it
- CHARGE OF THE PAYMENT: is the total amount to be paid by the user after a ride
- PAYMENT SYSTEM: is the external service that provides the payment functionality to the PowerEnJoy service
- APPLY A DISCOUNT: when at the end of a ride the charge of a payment is reduced according to discount policy of the service provider
- CHARGE --% EXTRA: when at the end of a ride the charge of a payment is increased of some percentage according to fine policy of the service provider
- POSITION: coordinates that localize something (a car, a user, some area…)
- USER EXITS THE CAR: is used as a synonym of end of 'end of the ride'
- END OF THE RIDE: when during a ride all the doors are closed and no more passengers are in the car
- AVAILABLE CAR: a car is said to be available when it is not unavailable (see definition of 'unavailable') and moreover it is possible for the users to reserve it

- UNAVAILABLE CAR: a car is said to be unavailable when it is reserved, being used by some user or with the battery level under 20%
- SAFE AREA: predefined set of places where it is possible for the users to park the cars
- CHARGING AREAS: are the parking provided with plugs to recharge the cars of the PowerEnJoy service (these are a subset of safe area)
- POWER GRID STATION: is a synonym of charge area
- STATE OFCHARGE: the state of charge of a car is the level in percentage of the energy stored in the battery
- PRE-DEFINED: when some information is supposed to have been previously provided to the database of the system ad therefore it is always available form the activation of the service
- WELL-DEFINED: when some information if defined precisely and without ambiguity
- JEB: this an acronym for Java Entity Bean

List of reference documents

During the writing of this document the following were consulted:

- Assignments AA 2016-2017
- The RASD previously produced
- The DD previously produced

Integration strategy

Entry criteria

Integration Test shall begin when the following criteria are met:

To find the defects in the module and verifies the functioning of software all the classes and methods must be tested thoroughly using JUnit. It is required that the test coverage of each class and package reaches 90% of the code lines.

The documentation for every method and class must be provided for each individual component, in order to make it easier to reuse classes and understand their functioning;

The System Operations and Administration Team has configured the Integration Test clients and servers for testing. The Test Team has been provided with appropriate access to these systems.

Along with the indications provided in this very document (ITPD), the two previous documents for this project, RASD and DD, must be delivered before the integration test phase can begin.

Elements to be integrated

This table below lists all the components that are going to be tested during the phase of integration testing.
The table was retrieved starting from the conceptual design made in the Design Document (section: architectural design -> high level components and their interaction -> conceptual design).

	Module name	Module description
Tiers	Client	This includes all types of clients the software can run on, which is to say the Mobile Application Client, the Web Browser Client and the On-Board Application Client, with all their internal components. The single client has to perform in accordance with its internal structure and has to be fully integrated with the other tier it interface with.
	Web server	This includes all the components (JSF) that interface with the Business Logic Tier, the ones that grant the comunication with the Client Tier and the ones that are in charge of the web interface
	Application server	This includes all the business logic for the application. All the interactions among internal components must be fully tested.
	Database	This includes the structures used to store the data, which is to say the DBMS and the Database Engine. The integration of the DBMS with the application logic must be fully tested.
Client explosion	Web browser	The Web Broser Client
	Mobile application	The Mobile Application Client
	On board computer	The OnBoard Computer Client

Session beans	User manager	This is the bean in charge of all the functionalities related to the management of the user information, such as the profile.
	Reservation manager	This is the bean in charge of all the functionalities related to the reservation of a car, starting from the request of display of all the cars in a certain area till the confirmation of the reservation.
	Pending reservation manager	This is the bean in charge of all the functionalities related to the active reservation, such as the unlock of the car or the cancellation of the reservation.
	Ride manager	This is the bean in charge of all the functionalities related to the active ride, such as charging the user and fetching the ending ride position of the car.
	CarStatus manager	This is the bean in charge of the modification of the cars statuses in the persistence layer.
	Payment gateway	This is the bean in charge of all the functionalities related to the interactions with the payment external system.
	Map manager	This is the bean in charge of all the functionalities related to the interactions with the Google Maps API.
Entity beans	User	This is the entity bean that stores user information
	Ride	This is the entity bean that stores ride information
	Reservation	This is the entity bean that stores reservation information

	Car	This is the entity bean that stores car information
	Payment	This is the entity bean that stores payment information
	Area	This is the entity bean that stores area information

Integration testing strategy

A bottom-up approach is the strategy that will be followed as regards the testing phase. This way, the testing can start from the smallest and the lowest-level components, that have already been tested at a unit level and don't depend on other components.
The objective of the Integration Test phase is to find bugs in the system under test, in the interfaces between components and in the system as a whole, respectively. We intend to do this by running a set of manual and automated test cases against each test release.
This strategy has been choose analysing different pros and cons.
Advantages:
- Advantageous if major errors occur toward the bottom of the program.
- Test conditions are easier to create.

- Observation of test results is easier.
- Number of stubs needed deeply reduced
Disadvantages:

- Driver Modules must be produced.
- The program as an entity does not exist until the last module is added.

At this level of integration testing, the communication functionalities with external systems must be covered as well, especially considering the relevance of said interaction in the context of the application. With respect to this, stubs and drivers will be used appropriately, based on the type of interface and interaction with the individual external systems.

Top-down analysis could be performed but it would bring these undesired disadvantages:
- Stub modules must be produced
- Stub modules are often more complicated than they first appear to be.

- Before the I/O functions are added, representation of test cases in stubs can be difficult.
- Test conditions may be impossible or very difficult to create.

- Observation of test output is more difficult.

- Allows one to think that design and testing can be overlapped.

- Induces one to defer completion of the testing of certain modules.

Sequence of component/function integration

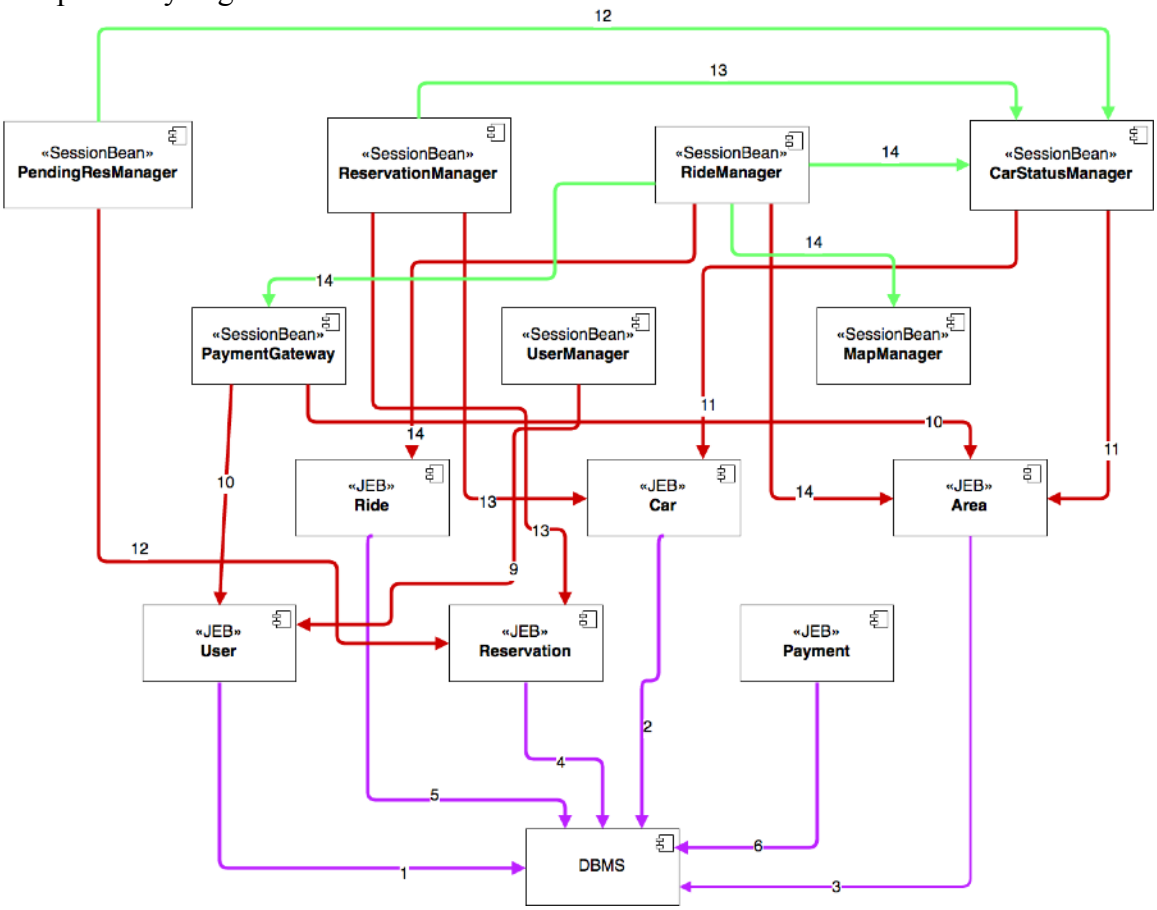
This section presents the ordering of the integration of components. The followed approach is to start from the most independent components to the least ones.

Software Integration Sequence

Here by means of the diagram we order the sequence in which the integration tests will take place. The blocks of the diagram represent the components and each arrows are related to an integration test. The numbers associated with the arrows represent the ordering.

The purple arrows represent the first step of the software integration sequence and they are related to integration between the JEB and the DBMS, we chose those first because they are supposed to communicate directly with the application data, which is the lower tier of the system.

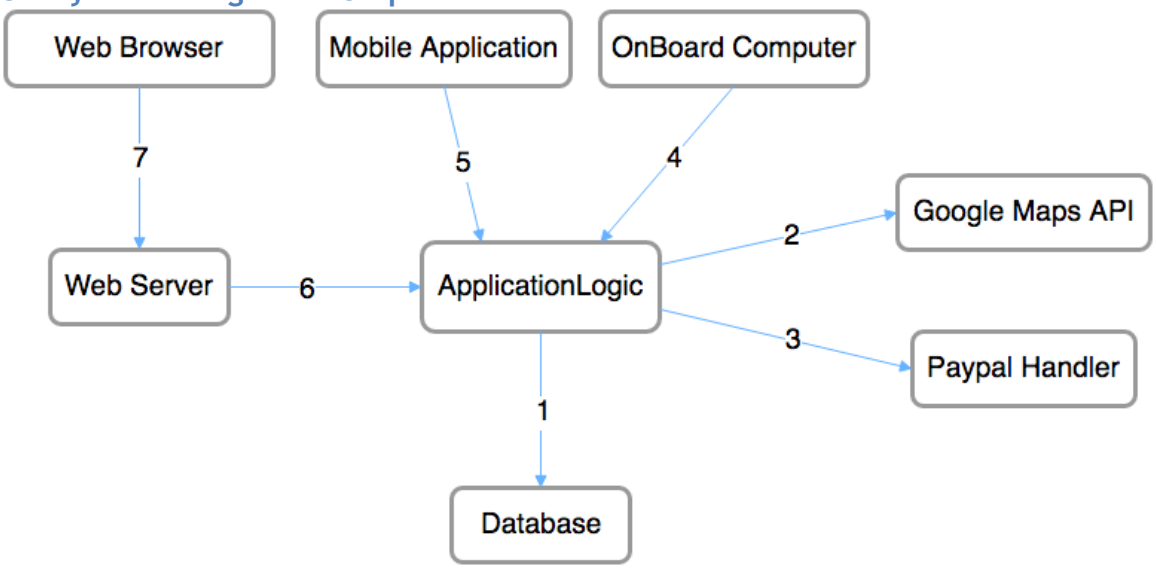
The red arrows represent the software integration between the Session Beans and the JEB, while the green arrows represent the software integration among the Session Beans. The ordering of this set of test is made according to the principle stated before of independency degree.



The following table lists all the integration test of the software without reflecting the ordering given above.

Identification number	Subsystem	Component	Integration test
IT1	Database, Application server	User (JEB)	-DBMS
IT2	Database, Application server	Ride (JEB)	-DBMS
IT3	Database, Application server	Reservation (JEB)	-DBMS
IT4	Database, Application server	Car (JEB)	-DBMS
IT5	Database, Application server	Payment (JEB)	-DBMS
IT6	Database, Application server	Area (JEB)	-DBMS
IT7	Application server	PendingReservation Manager	-CarStatusManager -Reservation (JEB)
IT8	Application server	UserManager	-User (JEB)
IT9	Application server	ReservationManager	-Reservation (JEB) -CarStatusManager -Car (JEB)
IT10	Application server	RideManager	-PaymentGateway -Ride (JEB) -CarStatusManager -MapManager -Area (JEB)
IT11	Application server	CarstatusManager	-Car (JEB) -Area (JEB)
IT12	Application server	PaymentGatewayManager	-Payment (JEB) -User (JEB)

Subsystem Integration Sequence



The first step in the subsystem integration process takes into consideration the ApplicationLogic and the Database, since the data tier is the most critic one as the whole set of components interact with it. Then we chose to test the external systems since they are characterized by the highest independency degree. Next we selected the OnBoard Computer and the Mobile Application integration because they are also supposed to interact directly with the ApplicationLogic. The Web Browser instead needs the Web Server to operate properly and therefore it's the last one component to be tested.

The following table lists all the integration test of the subsystem without reflecting the ordering given above.

Identification number	Subsystem	Integration test
IT13	Web browser	Web server
IT14	Mobile application	Application server
IT15	On board computer	Application server
IT16	Web server	Application server
IT17	Application server	Database
IT18	Application server	Paypal Handler
IT19	Application server	Google API Maps

Individual steps and test description

In this section we present one by one all the test set to be performed on the integration of each couple of components.

Test identification number	IT1
Test case description	Typical queries from the entity bean to the DBMS should be tested in order to verify that correct data are retrieved.
Items to be tested	
User JEB	DBMS
Input	Output
General queries for users data (login credentials, email, payment data,…)	Correct data must be returned

Test identification number	IT2
Test case description	Typical queries from the entity bean to the DBMS should be tested in order to verify that correct data are retrieved.
Items to be tested	
Ride JEB	DBMS
Input	Output
General queries for ride data (starting time, end time, ending position, total charge,...)	Correct data must be returned

Test identification number	IT3
Test case description	Typical queries from the entity bean to the DBMS should be tested in order to verify that correct data are retrieved.
Items to be tested	
Reservation JEB	DBMS
Input	Output

General queries for reservation data (reservation time, expiration time, car position,…)	Correct data must be returned

Test identification number	IT4
Test case description	Typical queries from the entity bean to the DBMS should be tested in order to verify that correct data are retrieved.
Items to be tested	
Car JEB	DBMS
Input	Output
General queries for car data (car status, car position, car soc,…)	Correct data must be returned

Test identification number	IT5
Test case description	Typical queries from the entity bean to the DBMS should be tested in order to verify that correct data are retrived.
Items to be tested	
Payment JEB	DBMS
Input	Output
General queries for payment data (total charge, payment date, payment state,…)	Correct data must be returned

Test identification number	IT6
Test case description	Typical queries from the entity bean to the DBMS should be tested in order to verify that correct data are retrived.
Items to be tested	
Area JEB	DBMS
Input	Output

General queries for area data (safe area, charging station,…)	Correct data must be returned

Test identification number	IT7.1
Test case description	The purpose of this test if to verify that when a user requests to cancel a reservation the PendingReservationManager correctly performs the task and calls the methods of the Reservation JEB to update the persistence data layer.
Items to be tested	
PendingReservationManager	Reservation JEB
Input	Output
-request reservation info -Cancel reservation	The task is performed correctly and the data layer is updated

Test identification number	IT7.2
Test case description	These test must simulate when a user wants to cancel a reservation and therefore the PendingReservationManager calls the methods of the CarstatusManager to request the update of the status of the car. The other scenario is when the user wants to unlock the car and the PendingReservationManager calls the methods of the CarstatusManager to request the update of the status of the car.
Items to be tested	
PendingReservationManager	CarstatusManager
Input	Output
-cancel reservation -unlock the car	The requests to the CarstatusManager are correctly performed

Test identification number	IT8

Test case description	The purpose of this test is to verify the possible calls of the UserManager on the User JEB that usually regard the update of the user information
Items to be tested	
UserManager	User JEB
Input	Output
-update user information	User info must be correctly updated and made persistent.

Test identification number	IT9.1
Test case description	The purpose of this test is to verify that when the user wants to reserve a car all the correct information are properly stored in the Reservation JEB
Items to be tested	
ReservationManager	Reservation JEB
Input	Output
-reservation info	The information related to the new reservation are correctly stored.

Test identification number	IT9.2
Test case description	These test must simulate when a user wants to confirm a reservation and therefore the ReservationManager calls the methods of the CarstatusManager to request the update of the status of the reserved car.
Items to be tested	
ReservationManager	CarstatusManager
Input	Output
-confirm reservation	The correct methods of the CarstatusManager are called

Test identification number	IT9.3
Test case description	This test verifies that when the ReservationManager wants to get the information about all the cars nearby a given position all data are correctly collected.
Items to be tested	
ReservationManager	Car JEB
Input	Output
-position around which locate cars	Correct cars data are returned to the ReservationManager

Test identification number	IT10.1
Test case description	The purpose of this test is to verify that the RdeManager correctly interacts with the PaymentGateway when it has to request a payment at the end of a ride.
Items to be tested	
RideManager	PaymentGateway
Input	Output
-payment data	Correct methods of the PaymentGateway are called

Test identification number	IT10.2
Test case description	This test verifies that the RideManager correctly calls the methods of the Ride JEB to create and update a ride istance.
Items to be tested	
RideManager	Ride JEB
Input	Output

-create a new ride -update a ride -terminate a ride	Data are correctly stored in the JEB

Test identification number	IT10.3
Test case description	This test verifies that the RideManager correctly calls the methods of the Castatus Manager in order to update the status of the given car.
Items to be tested	
RideManager	CarstatusManager
Input	Output
-update car status	The correct methods of the CarstatusManager are properly called

Test identification number	IT10.4
Test case description	This test verifies that the correct methods of the MapManager are called from the RideManager when it has to locate something.
Items to be tested	
RideManager	MapManager
Input	Output
-position request	Correct localization of the position is performed

Test identification number	IT10.5
Test case description	This test verifies that the RideManager correctly checks the properties of a given position checking it in the set of relevant areas (charging stations and safe areas).
Items to be tested	
RideManager	Area JEB

Input	Output
-position info	The correct properties of the position are retrieved.

Test identification number	IT11.1
Test case description	This test verifies that the CarstatusManager is able to correctly updates the information of a given car when required.
Items to be tested	
CarstatusManager	Car JEB
Input	Output
-new car status (isavailable, isreserved, islocked,…)	The information in the JEB are correctly updated

Test identification number	IT11.2
Test case description	This test verifies that the CarstatusManager correctly verifies that a car is in a correct position and eventually requires to some employee to move it.
Items to be tested	
CarstatusManager	Area JEB
Input	Output
-car position	Car position is correctly verified

Test identification number	IT12.1
Test case description	This test verifies that the PaymentGategay correctly stores and updates the information about payments.
Items to be tested	
PaymentGateway	Payment JEB
Input	Output

-payment info	Payment info are correctly stored or updated

Test identification number	IT12.2
Test case description	This test verifies that the PaymentGateway correctly interacts with the User JEB to retrieve the correct payment information of a given user.
Items to be tested	
PaymentGateway	User JEB
Input	Output
-payment info request	Correct payment information are retrieved

Test identification number	IT13
Test case description	In this test set should verify correct communication between the client and the Web server.
Items to be tested	
Web Browser	Web Server
Input	Output
General requests by the user	Correct web pages are shown

Test identification number	IT14
Test case description	In this test set should verify correct communication between the client and the application server.
Items to be tested	
Mobile application	Application server
Input	Output
General requests by the user	Correct information are provided to the mobile application

Test identification number	IT15
Test case description	In this test set should verify correct communication between the client and the application server
Items to be tested	
On board computer	Application server
Input	Output
General requests by the on board computer (update status, provide number of passengers,…)	Correct information are exchanged

Test identification number	IT16
Test case description	This test verifies that correct communication happens between the Web server and the Application server, also accomplishing the proper performance requirement.
Items to be tested	
Web server	Application server
Input	Output
General requests by the Web server	Correct response of the application server

Test identification number	IT17
Test case description	This test verifies that information requests are correctly performed and collected by the Application server
Items to be tested	
Application server	Database
Input	Output
General requests by the application server to the database	Correct information are provided by the database

Test identification number	IT18
Test case description	This test set verifies that the Appication server correctly performs the requests for payments to the PaypalHandler.
Items to be tested	
Application server	PaypalHandler
Input	Output
Payment request info	Payment is correctly performed

Test identification number	IT19
Test case description	This test set verifies that the Application server correctly calls the methods of the GooglemapsAPI to retrieve the position info.
Items to be tested	
Application server	GooglemapsAPI
Input	Output
Position info requests	Correct info positions are provided

Tools and test equipment required

This section lists and describes the software tools needed to accomplish the integration testing.

Test tools

Those listed are the software tools to be used during the integration test phase:

- MOCKITO: is a very simple tool to build the stubs required during the integration testing (in our case thanks to the bottom up approach those are many).
- ARQUILIAN: is a powerful tool to execute test cases for components. More precisely it allows simulating the containers that will constitute the actual environment in which those components have to work. In this way interactions between a component and its surrounding environment can be verified.
- JMETER: this tool can be used to simulate heavy load on servers. In our case we can apply them to all the layers underlying the client one: i.e. the web, business logic and data tier. The results related to those tests will be mostly related to how the system performs when managing a huge number of clients trying to access the service simultaneously.
- MANUAL TESTING: some part of the testing won't be automated and will require someone to actually try to use the service. Features mainly tested manually are the one related to the user experience.

To perform the tests some equipment will be required: for instance various smartphones that run different OS. In general almost all the platform that at the delivery of the project are supposed to be able to run the software system should be verified to properly support it.

Program stubs and test data required

In this section, according to the previously discussed testing strategy and planned tests, are listed the required stubs and drivers for each integration step.

A driver is a main program that creates an instance of the component to be tested in order to simulate in a simplified manner all the calls to the proper methods.
Drivers must be kept for future integration tests.
Because the two components involved in a specific integration test are already unit tested, the drivers that are made for them can be used in the integration test. This way testing can be performed more efficient.

A stub is a program that simulates a not yet developed component in order to allow the testing of some other one that is supposed to call methods in it.
As our approach to integration testing is bottom-up those tool will be required only in exceptional cases.

Test database
To perform proper integration tests a whole instance of the database is required. This database should be already populated with data that closely resembles production data so there will be no need to initialize this data or load additional data. Database scripts will be used for tables that need to be repopulated due to database structure changes.

Drivers for Java Entity Beans

This set of drivers is required to test the Entity Beans simulating both well-formed and malformed calls to the methods of those. Those calls have to make the Entity operate queries on the database in order to verify correct interaction.

Drivers for Session Beans

This set of drivers listed below is necessary for all the integration tests that involve a session bean. Both well-formed and malformed should be implemented.

PendingReservationManager

The driver for the PendingReservationManager session bean is supposed to call its methods in order to verify correct interactions with the CarstatusManager and the Reservation JEB.

UserManager

The driver for the UserManager session bean is supposed to call its methods in order to verify correct interactions with the User JEB.

ReservationManager

The driver for the Reservation session bean is supposed to call its methods in order to verify correct interactions with the Reservation JEB, the CarStatusManager session bean and the Car JEB.

RideManager

The driver for the ReservationManager session bean is supposed to call its methods in order to verify correct interactions with the PaymentGateway, CarStatusManager, MapManager session beans, Area JEB and Ride JEB.

CarStatusManager

The driver for the CarStatusManager session bean is supposed to call its methods in order to verify correct interactions with Car JEB and Area JEB.

PaymentGateway

The driver for the PaymentGateway session bean is supposed to call its methods in order to verify correct interactions with the Payment JEB and User JEB.

Divers and stubs for external systems interactions

In order to test the interactions of the system with the external components (PaypalHandler and GoogleMasAPI) both drivers and stubs are required.
The latter will be supposed to simulate the responses of the external systems to the requests of the Application server, the former to trigger those requests.
Both well-formed and malformed requests should be tested.

Effort spent

Madaffari Federico

- 02/01/17 (3h) group work
- 04/01/16 (2h)
- 05/01/16 (1h)
- 09/01/16 (1.5h) group work
- 10/01/16 (3h) group work
- 13/01/16 (3h) group work
- 15/01/16 (5h) group work

Mandrioli Claudio

- 02/01/17 (3h) group work
- 28/01/16 (3h)
- 05/01/16 (30m)
- 09/01/16 (1.5h) group work
- 10/01/16 (3h) group work
- 13/01/16 (5h) group work
- 15/01/16 (5h) group work