**DEMO**:

13/09/2016 TODO HOME

Demo scenarios:

1. Scenario local
2. Scenario docker local
3. Scenario docker Hub
4. Scenario PWS
5. Scenario Transactional
6. Scenario Eureka LOCAL
7. Scenario Eureka PWS

Sommario

[1.1 [REQUIREMENTS 1 – FUNCTIONAL ] 6](#_Toc463172199)

[1.2 [REQUIREMENTS 2– NON FUNCTIONAL ] 6](#_Toc463172200)

[1.3 [REQUIREMENTS 3– PROJECT MANAGEMENT ] 6](#_Toc463172201)

[1.4 [REQUIREMENTS FULFILLMENT – MICROSERVICES DATABASE PER SERVICE PATTERN ] 6](#_Toc463172202)

[Microservice and Database per service will fulfill the non functional requirements 6](#_Toc463172203)

[1.5 [REQUIREMENTS FULFILLMENT – TECHNOLOGY STACKS] 6](#_Toc463172204)

[1.6 [REQUIREMENTS FULFILLMENT – LIFECYCLE ] 7](#_Toc463172205)

[1.7 [SYSTEM LANDSCAPE] 8](#_Toc463172206)

[2 Demo 9](#_Toc463172207)

[2.1 [DEVELOPMENT / UNIT TEST ] 10](#_Toc463172208)

[2.1.1 LOCAL H2 11](#_Toc463172209)

[2.1.2 LOCAl MYSQL 11](#_Toc463172210)

[2.1.3 [SCENARIO INTEGRATION TEST/DOCKER] 12](#_Toc463172211)

[2.1.4 [SCENARIO QUALITY ASSURANCE–DOCKER HUB] 13](#_Toc463172212)

[2.1.5 [SCENARIO PRODUCTION–PIVOTAL WEB SERVICES] 14](#_Toc463172213)

[2.1.6 [EVENT DRIVE DEMO] 15](#_Toc463172214)

[3 Wiring microservices - Service Discovery 17](#_Toc463172215)

[3.1.1 [CONCLUTION] 17](#_Toc463172216)

**[INTRODUCTION]**

Welcome everybody my name is Luigi Bennardis, I’m an information Technology architect with a seniority of about twenty years. Actually I’m in charge of the design of the ALM platform for the Italian Postal Services.

**[PRESENTATION]**

A the title suggest the aim of this project is to provide a digital platform based on a microservices architecture, which will require an application lifecycle management process.

The gathering of the requirements will be the first phase of the process, followed by the choice of the technical design.

The third step is the development of the software and its integration.

The fifth phase is the quality assurance, followed by the sixth and final phase: delivery.

architectural (database per service) and technological (Spring Boot) aspects to delivery related scenarios (development, Cloud or dockerized environments), in an ecosystem context where microservices are each other reliable and fault tolerant (Eureka service registry, Ribbon load balancing, Spring Cloud).

**[FIRST IMAGE OF THE MAP OF ROME – HISTORICAL DISTRICT]**

This is historical district of Rome.

It is a very ancient area whose urban development dates back to the roman empire, more than  2000 years ago, through the ages, when the city structure has not evolved in a sustainable way.

**[SECOND IMAGE – HISTORICAL DISTRICT OF ROME]**

The blue line surrounds the limited traffic zone .

It is an area of about 10 square kilometers caracterized by a strong concentration of historical monuments, tourism facilities and national institutions, government and parliament among others, that generate a consistent demand for mobility.

As a matter of a fact the urban development was not planned originally planned for the actual need of people and motor vehicle mobility. The street are very often often too narrow in a context of a fragile architectural environment.

So the current mobility situation of this area is unsustainable.

**[THIRD IMAGE – SMART ELECTRIC VEHICLE]**

The solution to this mobility problem ia s network of smart electrical vehicles with removal batteries whose organization is powered by a microservices based digital platform.

In this way we colud create a modern urban mobility network thata is both economically and environment sustainable.

[FOURTH IMAGE – SMART ELECTRIC VEHICLE – BATTERY ]

To overcome the limited life of the batteries these vehicles has been engineered with a special battery pack that could be easily and rapidly changed in dedicated service station.

In this way this vehicles will not need to stop to charge theirs batteries and so could be much more efficient in comparison with the other electric vehicles.

[FIFTH IMAGE – STATION ]

I do not know if this is a realizable scenario but if so it is necessary a software solution by means manage it.

The requirements are that Every driver according to his delivery plan program each morning the expected battery changes.  
This by means of a mobile application he will book one or  more pit stops for a fresh batteries supplied by the stations.

Besides in case of emergency it will be the vehicle itself that will notify to the driver the necessity to come to the nearest station with available batteries.

So these vehicle could lead an ideal  24/7 service without charging stops in comparison with those vehicles not engineered in such a way.

## [REQUIREMENTS 1 – FUNCTIONAL ]

## [REQUIREMENTS 2– NON FUNCTIONAL ]

## [REQUIREMENTS 3– PROJECT MANAGEMENT ]

## [REQUIREMENTS FULFILLMENT – MICROSERVICES DATABASE PER SERVICE PATTERN ]

## Microservice and Database per service will fulfill the non functional requirements

So a Microservices architecture will be the design pattern that accomplish the listed requirements (face the choose of a database design pattern)

What is the database design pattern that best fit a microservice architecture?

Description of database per service pattern for developing Micorservices

Database per service will fulfill the non functional requirements

## [REQUIREMENTS FULFILLMENT – TECHNOLOGY STACKS]

A possible solution of these requirements could be led by the following technology stack coming by the stìpring project abd Netflix

A possivle resolution of these requirement could be led by The follpwimg  texhnology stack spring and netflix based .

In this chart i have tried to summarize the technology stacks that I have used to realize the design patterns defined for choose for the development.

In this chart I have tried to summarize the technology stacks that realize the design patterns choosed for the architecture of this system

SPRING BOOT RUNNABLE FROM COMMANDLINE EXAMPLE

In this chart i have posizionato the    that accomplish the lifecycle steps imagined for this project and the corrispondending realizing services  
both in a local and in a cloud environment

## [REQUIREMENTS FULFILLMENT – LIFECYCLE ]

In this chart I have arranged that will accomplish the lifecycle steps imagined for this project and the corresponding realizing services both in a local and in a cloud environment.

In this chart i have arraged the    that accomplish the lifecycle steps imagined for this project and the corrispondending realizing services   
both in a local and in a cloud environment.

Lifecycle vs service vs env  
In this chart i have tried to summarize the technology stacks that realize the design patterns chosed for the development.

## [SYSTEM LANDSCAPE]

Face the design of one of the microswrvices of the system. That which records the booking information.

Face the design of one of the microservice of the system that which record the booking information

* Show picture (db/logic/api)

This is the simplest landscape of a microservices architecture. There is a simgle service who expose his business logic only by means of restfull web service api. During this speech i will add new seevices that realize other aspects of the system.

The landscape of this system will became muxh more real during this presentation adding new services or microservices talking about event driven architecture and wiring microservices

# Demo

**DEVELOPMENT::PRELOAD CHROME PAGES**

DEV\_LOCAL.bat

**INTEGRATION::PRELOAD**

1. **COMMAND LINE DOCKERMACHINE RUN**
2. **COMMAND LINE ready to run list of container**
3. **COMMAND LINE READY TO RUN CONTAINERS** 
   1. **MYSQL**
   2. **APP WITH LINK**
4. **CHROME PAGES**

**DOC\_LOCAL.bat**

**QUALITY ASSURANCE::PRELOAD CHROME PAGES**

DOC\_HUB.bat

**PRODUCTION::PRELOAD CHROME PAGES**

**EVENT DRIVEN::PRELOAD CHROME PAGES**

EVENT.BAT

**CHECK DOCS**

## [DEVELOPMENT / UNIT TEST ]

**INTRODUCTION TO THIS PHASE**

Besides the implementation details of the frameworks in this first phase of lifecycle, development,

We see the resolution if the database backing service

Keeping the same coding both for a local in memory He , MySql

* Spring boot and services binding
  + Automatic behaviuor in cloud foundry
  + The goal should be a smooh transition from local execution form developer desktop to production in Cloud Foundry
  + Binding realized with declarative approach
    - Application.properties
    - Yaml file
    - [show the differences]
    - Dual Running: Local with H2/mysql
    - And the in the Cloud with MySQL
    - Configuration
  + Different file
* H2 in memory database
* <https://spring.io/guides/gs/accessing-data-rest/>
* <https://spring.io/guides/gs/accessing-data-jpa/>
* <http://blog.netgloo.com/2014/10/27/using-mysql-in-spring-boot-via-spring-data-jpa-and-hibernate/>

**[SHOW PPT SLIDES WITH IMPLEMENTATION DETAILS]**

### LOCAL H2

00pwd PROFILE h2

Environment information provided by spring boot

<http://localhost:7111/h2-console/>

<http://localhost:7111/trace>

<http://localhost:7111/flyway>

<http://localhost:7111/metrics>

example /flyway number of request

<http://localhost:7111/bookABattery/list>

list of booked batteries – EMPTY

<http://localhost:7111/bookABattery/addBooking/STAZ001/BATT011/ROMA/42.984545454/16.74444/>

add one or more booking

<http://localhost:7111/bookABattery/list>

list of booked batteries – LIST EVIDENCE

<http://localhost:7111/findNearestStation/41.90231/12.4832/0.50/>

find nearest station for an emergency pit stop

### LOCAl MYSQL

00pwd PROFILE mYSQL

[**http://localhost:7111/health**](http://localhost:7111/health)

**show the Jason with information about the disk the up status and the database in this cas MySQL**

<http://localhost:7111/bookABattery/addBooking/STAZ001/BATT011/ROMA/42.984545454/16.74444/>

add one or more booking

<http://localhost:7111/bookABattery/list>

list of booked batteries – LIST EVIDENCE

<http://localhost:7111/findNearestStation/41.90231/12.4832/0.50/>

find nearest station for an emergency pit stop

### [SCENARIO INTEGRATION TEST/DOCKER]

**INTRODUCING**

**now build a Docker image that runs the Spring Boot application.**

**Introducing a scenario of shiiping the application inside a container**

**THE INTEGRATION TEST OF A COMPONENT MUST REFERENCE / USE OTHER COMPONENT S THAT COMES FROM OTHER DEVELOPING LINES**

**SO THE DOCKER REGISTRY SHOUD BE THE HUB OF ALL THE MICROSERVICESOF THIS DIGITAL PLATFORM**

**IT WILL NOT NEE TO MANAGE DEPLOYMENT OF EXECUTALE ARTIFACTS, ONLY PULL THE IMAGE NEEDED AND EXECUTING LINKED TO ONE ON WHICH WE WANT TO CONDUC INTEGRATION TEST**

**Special images with data specific for test**

**Shiiping a inmage rather tha executable software artifact**

**artifact to deployed in qa machine**

**There is no limit to crate dedicated qa/integration machine**

**so it will be superated the sizing and the procurament a priori of machine**

**So let’s go ahead in the develipong process facing the phase of tests of integration test**

**The commit master will have the duty of conduct the integration test against all development done on one or more sevices modified for the new system release or for bug fixing coming from the QA or production environments.**

**THE INTEGRATION TEST WILL BE CONDUCTED IN AN EVIRONMENT WHERE THERE WILL BE AVAILABLE ALL THE SERVICES needed for the system to be consistent.**

**In a simple scenario two or more developers each responsible for the development of a part of a microservice finish with success the unit test**

**Beside the skill of the qa team must be enough to get the application and run it, it is stated in the requirements that in this team there will be no need of technical slìkill that will manage the deployment or other technical tasks.**

**So the commit master in a dedicated branch deliver the work of the developers merging all the contributes and build the microservice**

Docker definition

-I have provided an empty database image that will be load by flyway at start up

Show Dockerfile

Show dockerfile template

Show maven implementation and plug in

Create a local docker image

Run the container + database

Show bindings

**BEFORE FIRST SLIDE**

**I’m going to represent some higlights of configuration, mainly on the maven build plugins and the dockerfile.**

### [SCENARIO QUALITY ASSURANCE–DOCKER HUB]

**INTRODUCTION**

Among the requirements it has been stated that the project’s teams

The progect management requirements shoud accomplish teams geographically distributed.

In such a context the delivery of the application as a docker machine could be realized by means of Docker Hub as a project’s images registry.

So let’s go ahead in the develipong process facing the phase of QUALITY ASSURANCE TEST

The commit master will have the duty of conduct the integration test against all development done on one or more sevices modified for the new system release or for bug fixing coming from the QA or production environments.

THE QA TEST WILL BE CONDUCTED IN AN EVIRONMENT WHERE THERE WILL BE AVAILABLE ALL THE SERVICES needed for the system to be consistent.

In a simple scenario two or more developers each responsible for the development of a part of a microservice finish with success the unit test

Beside the skill of the qa team must be enough to get the application and run it, it is stated in the requirements that in this team there will be no need of technical slìkill that will manage the deployment or other technical tasks.

So the commit master in a dedicated branch deliver the work of the developers merging all the contributes and build the microservice

DOCKER REGISTRY WITH ALL THE IMAGES OF THE MICROSERVICES OF THE DIGITAL PLATFORM

GIT HUB A AS AMAVEN REPOSITORY OF ALL ARTIFACTS EVETUALLY NEEDED BY THE PROJECT

### [SCENARIO PRODUCTION–PIVOTAL WEB SERVICES]

**DESCRITPION**

INTRODUCE ANOTHER PAAS PROVIDER FOR THIS DIGITAL PLATFORM

Quality assureance test has finished successfully so the next of the lifecycle is Production

For the runtime environment it will be used a Cloud PAAS : Pivotal Web Services

Pivotal Web Services Built on Cloud Foundry AN OPEN SOURCE paas sOLUTION

* + Backing services
  + Yml file
  + Create from interface
  + By Manve and then integrate to Jenkins
  + BY cf tool

Database as a Service Pattern implemented with a dedicated instance

Database as service pattern will enforce database per service pattern.  Dedicated instance , dedicated schema and horizontal scale up

Show definition of application .cloudfoundry and its link to application.yml

Deploy by cf tool – maven (only show plugin)

Pws  
Database as service pattern will enforce database per service pattern.  Dedicated instance , dedicated schema and horizontal scale up

Database per service  pattern implementation inside a cloud env is natively implemented  
Segregation is enforced

The databasr is dedicated to the servuce and could be scaled out independently

### [EVENT DRIVE DEMO]

**INTRODUCTION**

Event driven  
Example if extra coding and infrastructure management.   
Show the effective overhead of coding and infra management in a simple example.  
Drawback  
Benefits

# Wiring microservices - Service Discovery

INTRODUCTION

In such an architecture microservices can cooperate for a computing system ecosystem shoud wire each other

Wiring microservices.  
Loosing coupled requirements are not onky those involving tech aspects. In a ecosystem if microservice it is likely thata microservices need each othe wiring.  
Microservices do nita act by themself.  
Si the loosing coupled requirent is completely satisfied implementibg a service discovery a registry where are resolved........

Wiring by resftfull call baut to achieve decouplig it is necessary that service shoud be discocered.

Eureka server is on its turna microservice (run as a spring boot application)

### [WHAT’S NEXT]

### [CONCLUTION]

So that’s all, wishing that this presentation beeing clear enought has catched your , I would like to thanks you of your attention and O’Reilly for the opportunity given to me to take part at such an important event like that