Tile-in-ONE

An integrated framework for the data quality assessment and database management for the ATLAS Tile Calorimeter

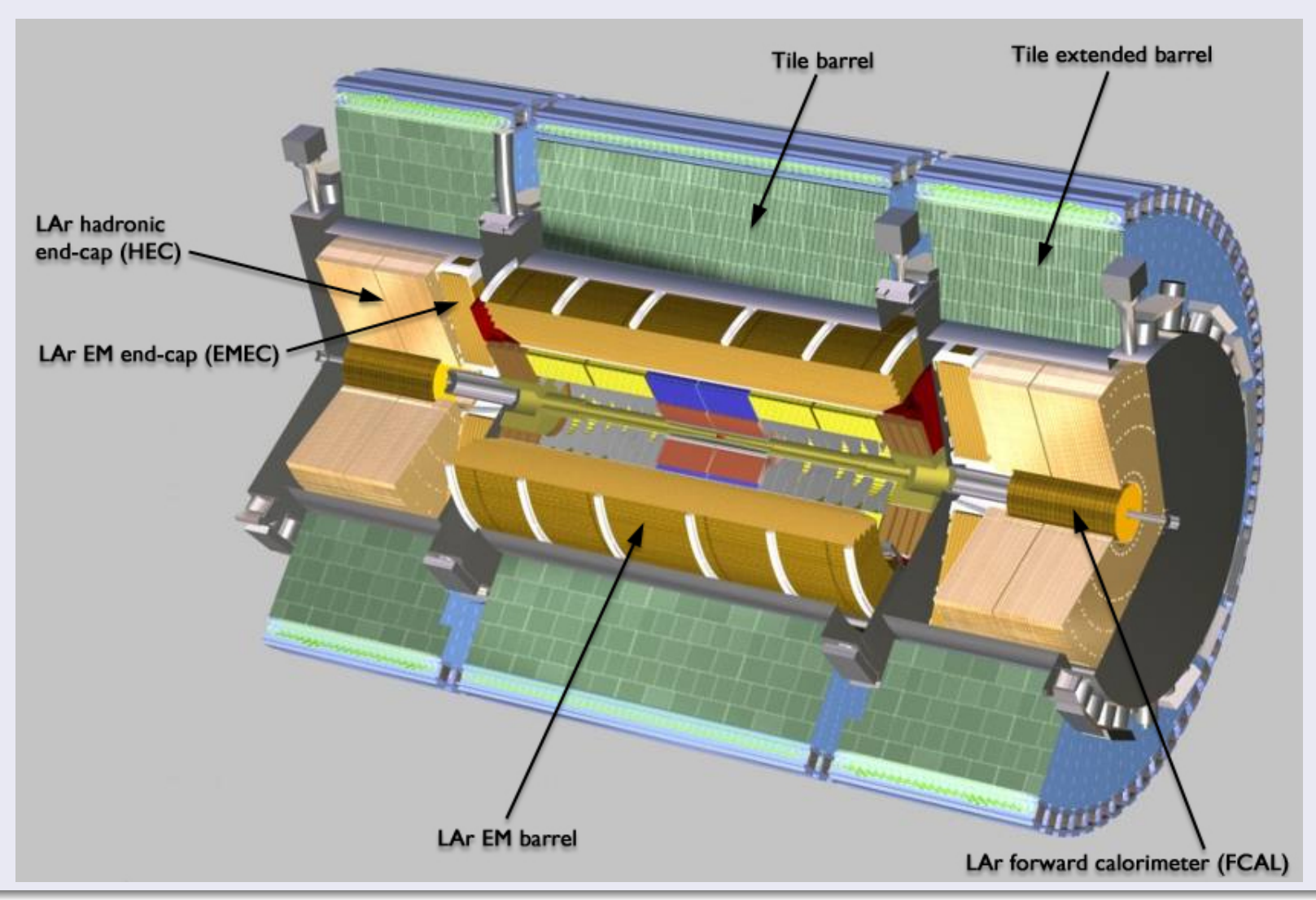
Raffaela Cunha [1] , Carlos Solans [2] , Andressa Sivolella [1] , Fernando Ferreira [1] , Carmen Maidantchik [1]

(1) Universidade Federal do Rio de Janeiro (UFRJ), Brazil
(2) Organisation européenne pour la recherche nucléaire (CERN)
Mail: tile-in-one@cern.ch

International Conference on Computing in High Energy and Nuclear Physics
October 14-18, 2013 Amsterdam, The Netherlands

Introduction

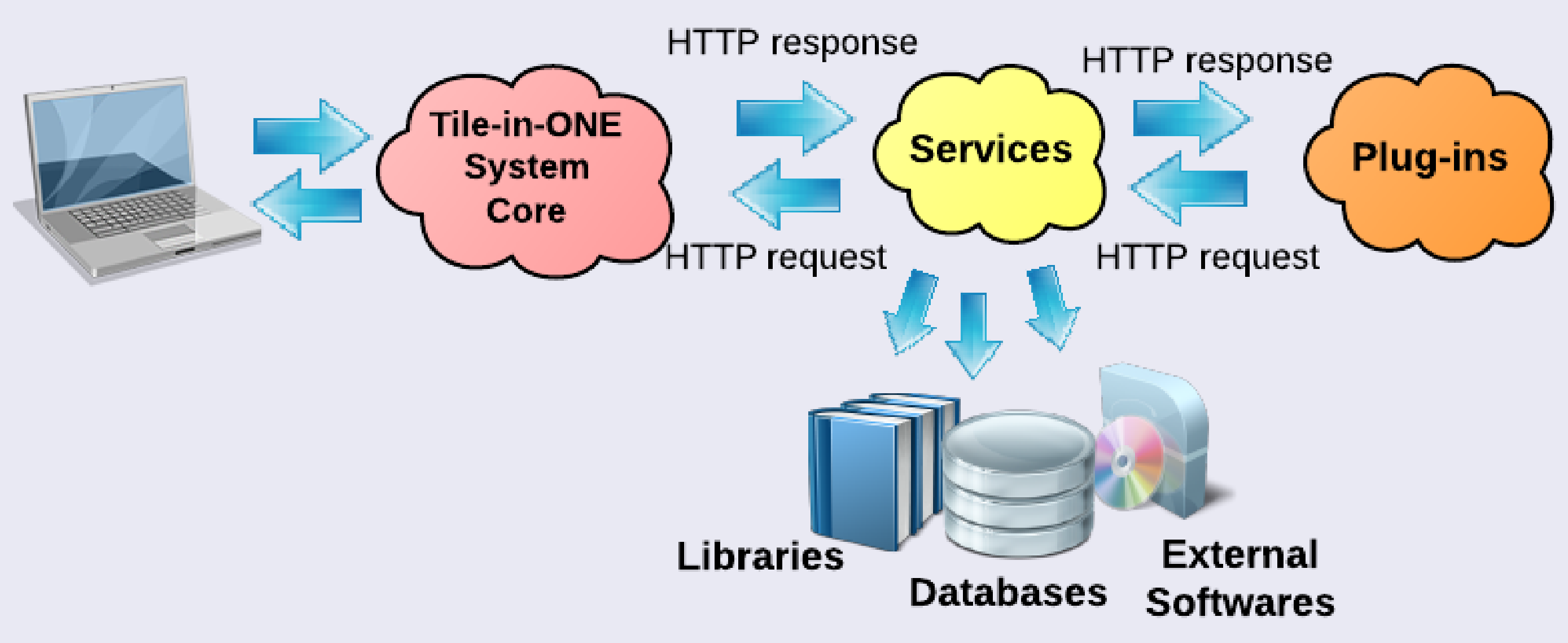
The Tile calorimeter is one of the sub-detectors of ATLAS. In order to ensure its proper operation and assess the quality of data, many tasks have to be performed by means of different tools which were developed independently. Thus, these systems are commonly implemented without a global perspective of the detector and lack basic software features. Besides, in some cases they overlap in the objectives and resources with another one.
Tile-in-ONE is an infrastructure designed to integrate these tools, making them able to access common services and be displayed in a single interface. Moreover, collaborators should be allowed to develop their own tools exploiting any existing feature.



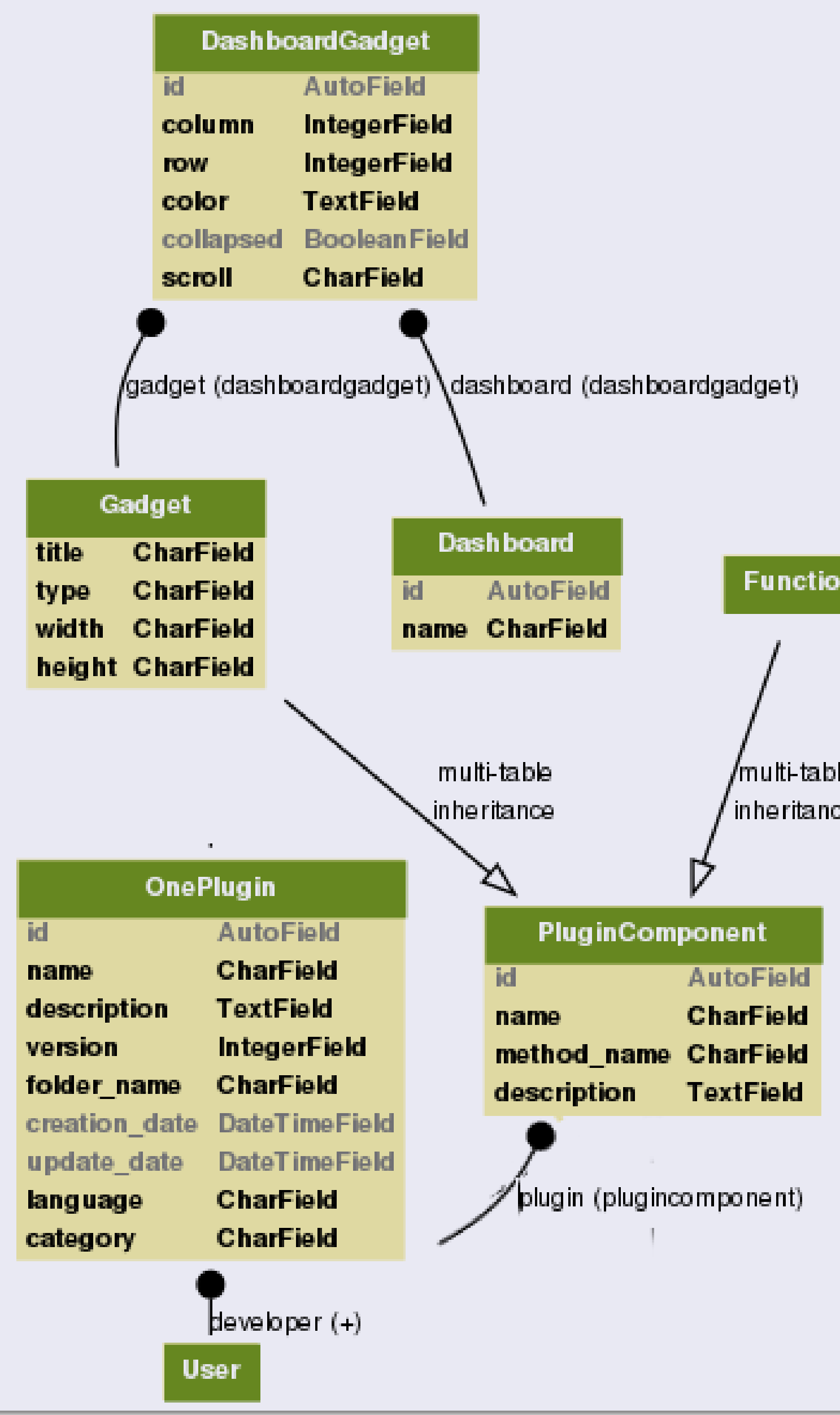
The system architecture

The system is composed of three basic elements:

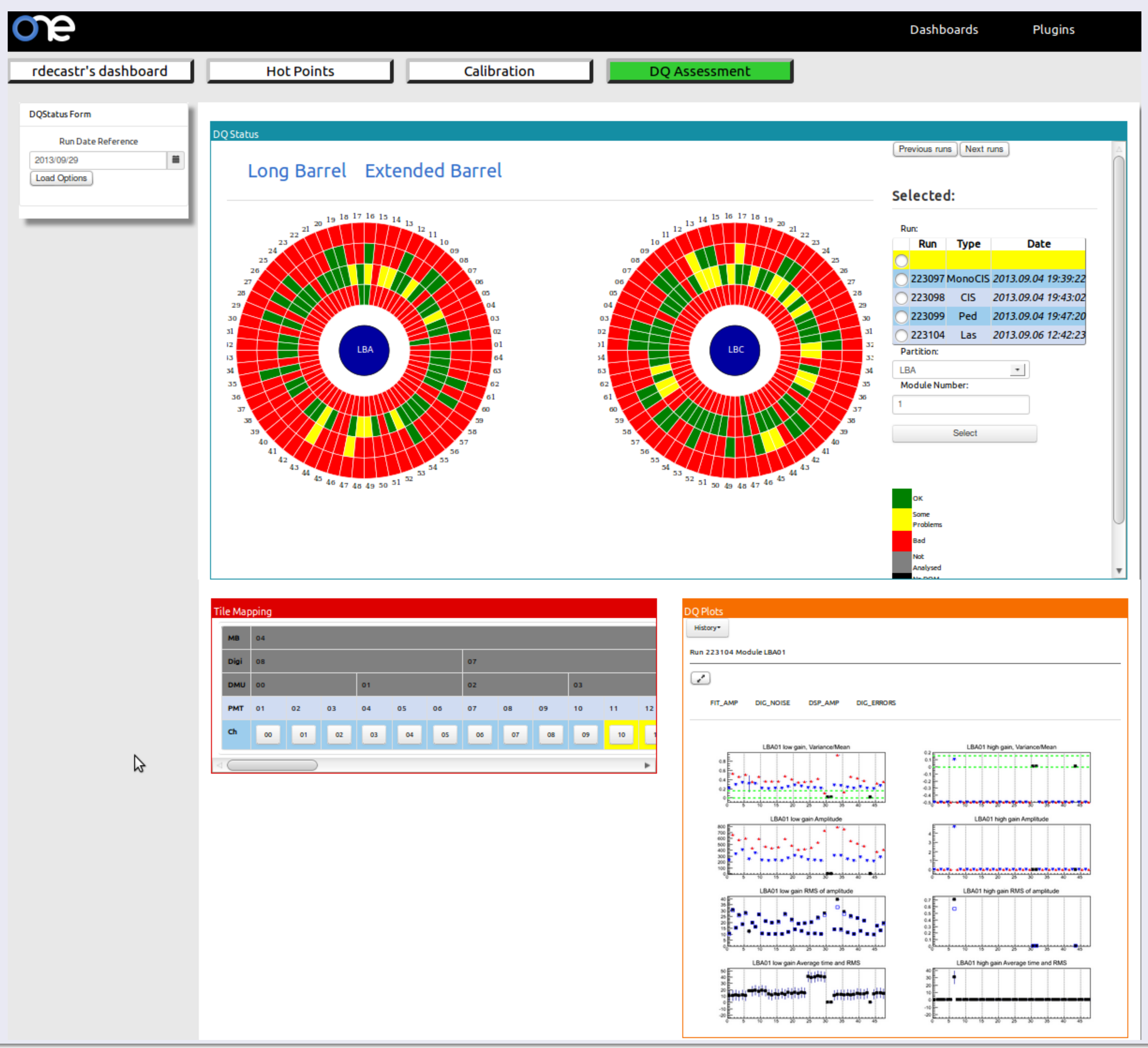
- The system core: represents the basic framework that loads the configuration, manages user settings and loads plug-ins at runtime.
- The services: provided to centralize the access to the resources. A specific service allows the system core to access any plug-in as well as a plug-in to access any other.
- The plug-ins: it is a set of components designed to provide some functionality for the final user.



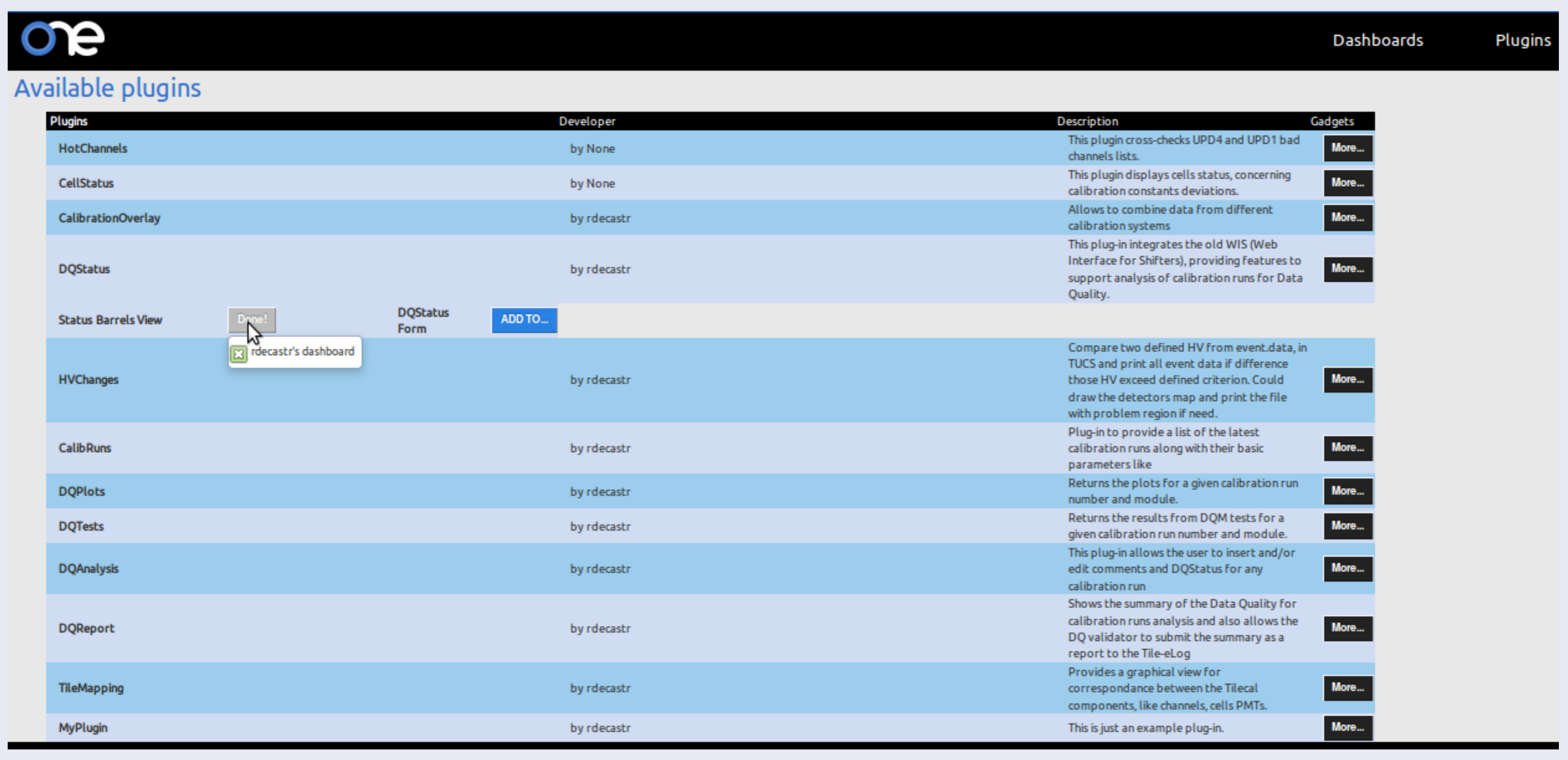
The access from the system core to the plug-ins and from the plug-ins to the resources are made by sending an HTTP request for a given service.
A component which returns some content to be displayed in the interface is defined as a "Gadget". The developer can also define a component as a "Function", to make it able to be called by other plug-ins through a mediator service.



The web interface - Dashboards and Gadgets



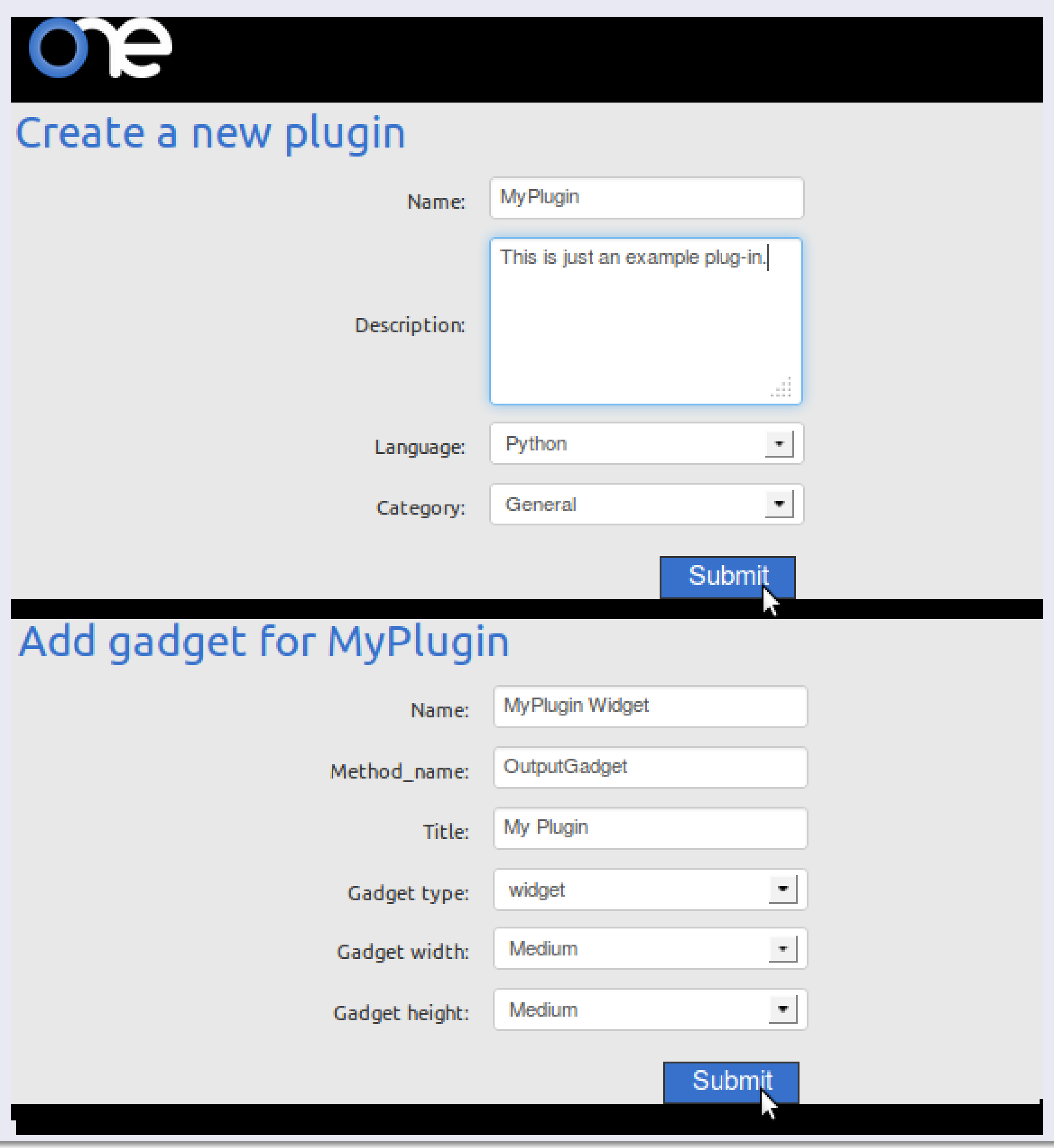
The first time a user logs into the system, a dedicated dashboard is created. It is possible to add any of the available plug-ins into the user's dashboard. Some plug-ins are available for dedicated expert role. Other dashboards exist in the system which are designed for a specific purpose. For example, the "DQ Assessment" provides a pre-defined layout of the required plug-ins to assess the quality of the data. In this dashboard, the user is presented with a set of calibration runs displayed as concetrinc rings, each wedge representing a module. With a simple view the user can compare the status of the module across the different runs. This view will also display results from physics runs if any.



Available plugins

Plugins	Developer	Description	Gadgets
HotChannels	by None	This plugin cross-checks UPD4 and UPD1 bad channels lists.	More...
CellStatus	by None	This plugin displays cells status, concerning calibration constants deviations.	More...
CalibrationOverlay	by rdecastr	Allows to combine data from different calibration systems	More...
DQStatus	by rdecastr	This plug-in integrates the old WIS (Web Interface for Shifters), providing features to support analysis of calibration runs for Data Quality.	More...
Status Barrels View Done! rdecastr's dashboard	DQStatus Form ADD TO...		
HVChanges	by rdecastr	Compare two defined HV from event.data, in TUCS and print all event data if difference those HV exceed defined criterion. Could draw the detectors map and print the file with problem region if need.	More...
CalibRuns	by rdecastr	Plug-in to provide a list of the latest calibration runs along with their basic parameters like	More...
DQPlots	by rdecastr	Returns the plots for a given calibration run number and module.	More...
DQTests	by rdecastr	Returns the results from DQM tests for a given calibration run number and module.	More...
DQAnalysis	by rdecastr	This plug-in allows the user to insert and/or edit comments and DQStatus for any calibration run	More...
DQReport	by rdecastr	Shows the summary of the Data Quality for calibration runs analysis and also allows the DQ validator to submit the summary as a report to the Tile-eLog	More...
TileMapping	by rdecastr	Provides a graphical view for correspondance between the Tilecal components, like channels, cells PMTs.	More...
MyPlugin	by rdecastr	This is just an example plug-in.	More...

Plug-ins API and development



Create a new plugin

Name:	MyPlugin
Description:	This is just an example plug-in.
Language:	Python
Category:	General

Submit

Add gadget for MyPlugin

Name:	MyPlugin Widget
Method_name:	OutputGadget
Title:	My Plugin
Gadget type:	widget
Gadget width:	Medium
Gadget height:	Medium

Submit

In order to create a plug-in, the developer should:

- specify the plug-in name, description and the programming language to be used
- define one or more gadgets to be displayed and/or functions to be reused
- in case of python language, write the plug-in code as a class with the same name as the plug-in and methods which return the content to be displayed in the specifyed gadgets or functions. If needed to use some service, a function named "callService" should be called, specifying the name of the service, the method and parameters to be sent.

```
class MyPlugin(OneBasePlugin):
    def __init__(self, params):
         self.params=json.loads(params)
         self.bar_name=self.params['bar_name']
         self.ch_number=self.params['ch_number']
         self.mod_number=self.params['mod_number']

    def Request(self):
        module=self.bar_name+self.mod_number
        params={'module':module,'channel':self.ch_number}
        cur_cell=callService("TilecalMapping","GetCellByChannel",params)
        cur_pmt=callService("TilecalMapping","GetPMTByChannel",params)
        result_list=[cur_pmt[0],cur_cell[0]]
        return result_list

    def OutputGadget(self):
        data=self.Request()
        html += "<h3>Result</h3>"
        html+="<p>PMT:"+ data[0]
        html+="</p><p>Cell:"+data[1]
        html+="</p>"
        return html
```