Hue Application for Big Data Ingestion

August 2016

Author:
Medina Bandić

Supervisor(s):
Antonio Romero Marin
Manuel Martin Marquez

CERN openlab Summer Student Report 2016

Abstract

The purpose of project was to develop a web application for the HLoader - a data ingestion and ETL framework developed at CERN. This framework automates data streaming/ingestion from a wide range of data services into CERN Hadoop Big Data analytics cluster. The web application uses the HLoader REST API to interact with the framework and expose the ETL functionalities to the users.

This report will introduce the HUE framework for web applications and explain why it is useful and how to use it. The report will give an overview of the frameworks and the technologies used for the project. This report will also show how to create an application in HUE and use the HLoader REST API. Finally, it will present the project future plans and improvements.

Table of Contents

1 Introduction

Big Data ingestion is a very complex process. Especially when we speak about data ingestion in CERN Hadoop clusters where a great amount of data is ingested from various sources like experiments, accelerators, systems, infrastructure, services and so on. That implies the necessity of simplification of the process and easier access to perform efficiently the tasks required during the process.

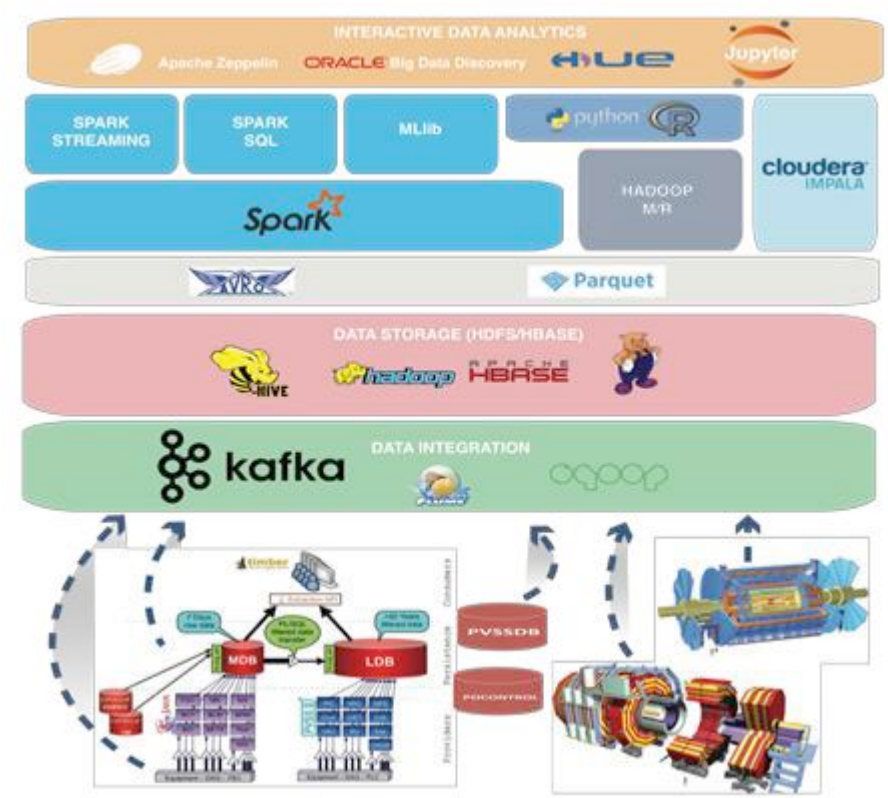

Figure 1: Big Data Ingestion

The purpose of this project is to create a web interface to interact with the HLoader framework developed at CERN to simplify the ETL processes for Hadoop. This web application should facilitate the interaction with the HLoader framework, improving the ETL (extract–transform–load) process for both users and service administrators. For this purpose, the HUE framework offers multiple features that can be used by the web application.

2 HUE Framework

2.1 Why HUE?

HUE - which stands for 'Hadoop User Interface' - is a web application built in Django framework and placed between Hadoop and browser. It hosts built-in and user HUE apps and provides an environment for analyzing and exploring data.

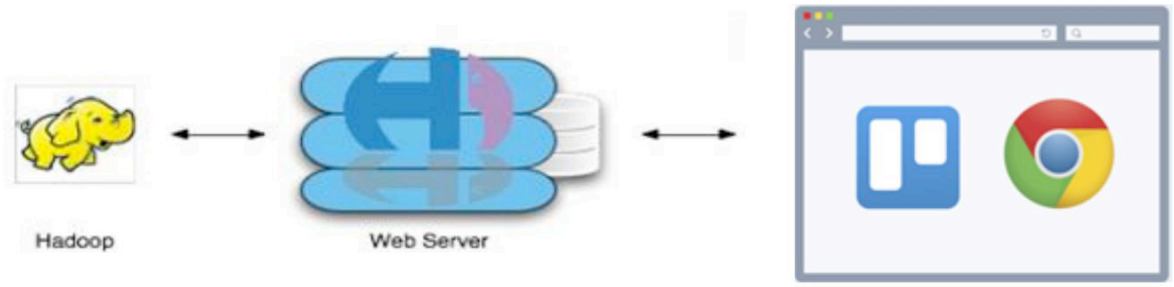

Figure 2: HUE

There are various reasons to choose this framework in this use case. First if all, it is already integrated with Hadoop system. Most noticeably, there are python file-object-like APIs for interacting with HDFS. These APIs work by making REST API or Thrift calls to the Hadoop daemons. The Hadoop administrators must enable these interfaces from Hadoop.

Considering that this project is for automating data ingestion in Hadoop clusters, that feature is huge benefit. It also supports different SQL editors for Hive, Impala, MySQL, Oracle, PostgreSQL, SparkSQL, Solr SQL and Phoenix, as well as scheduling of jobs and workflows through an Oozie Editor and Dashboard. Other benefits of using this framework are: collaborative user interface, configuration management, supervision of sub-processes, user management, integrated security and interaction with other applications.

2.2 HUE Architecture

A HUE application has three layers. The first one is the front-end for user interaction which includes usage of jQuery, Bootstrap and HTML/CSS. The second layer is the overall application logic contained in its web server. Basically, the Django framework is for that part and Python as main language. The third layer includes external backend services with which the application is interacting.

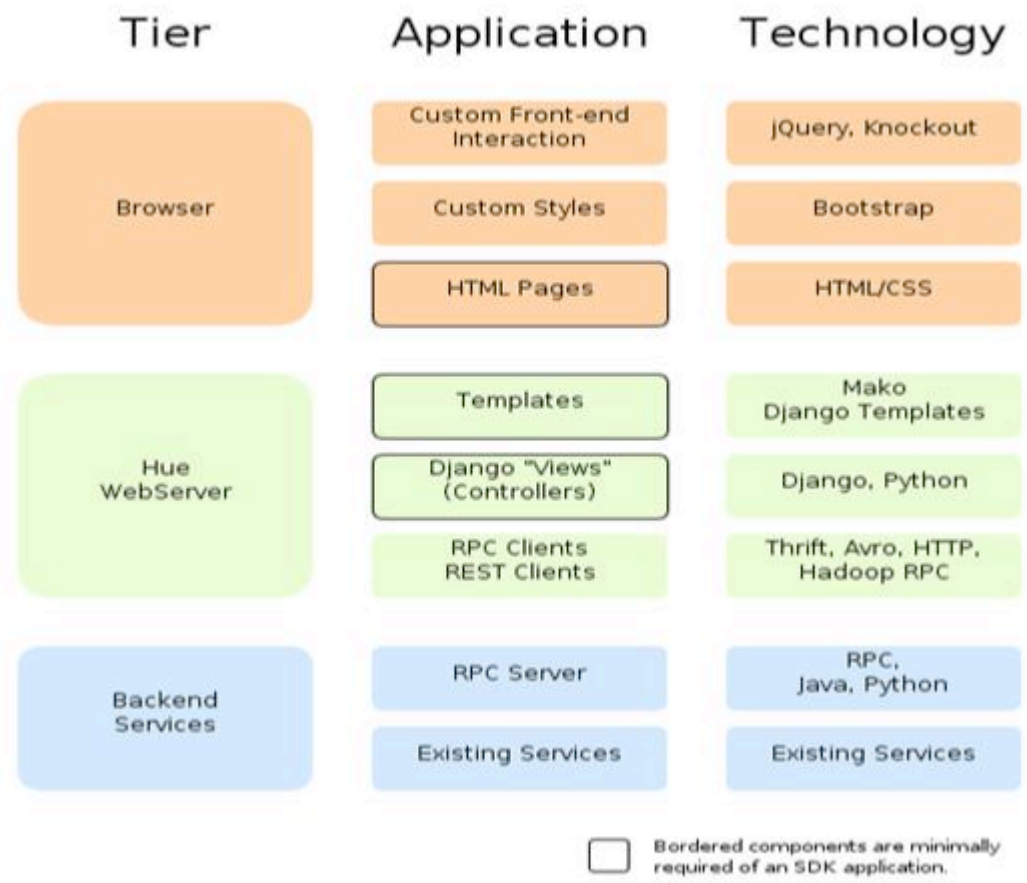

Figure 3: HUE Architecture

The minimum that should be implemented is a "Django view" – controller function that processes the request and the associated template to render the response into HTML. Except for the web server, Hue applications run some other daemon processes. That is the preferred way to manage long-running tasks for which might exist together with the web page rendering. This communication between views and daemons is done by using Thrift or by exchanging state through the database.

2.2.1 Django Framework

Django is an MVC (model-view-controller) web framework. The only difference is that controller is called “view” and a view is called “template”. Django is running on WSGI container/web server, manages the URL dispatch, executes application logic code, and puts together the views from their templates. Python is used for the application logic and HTML or Mako templates for user interface. Django views accept requests from users and perform the necessary changes on models, which represent the data storage of application.

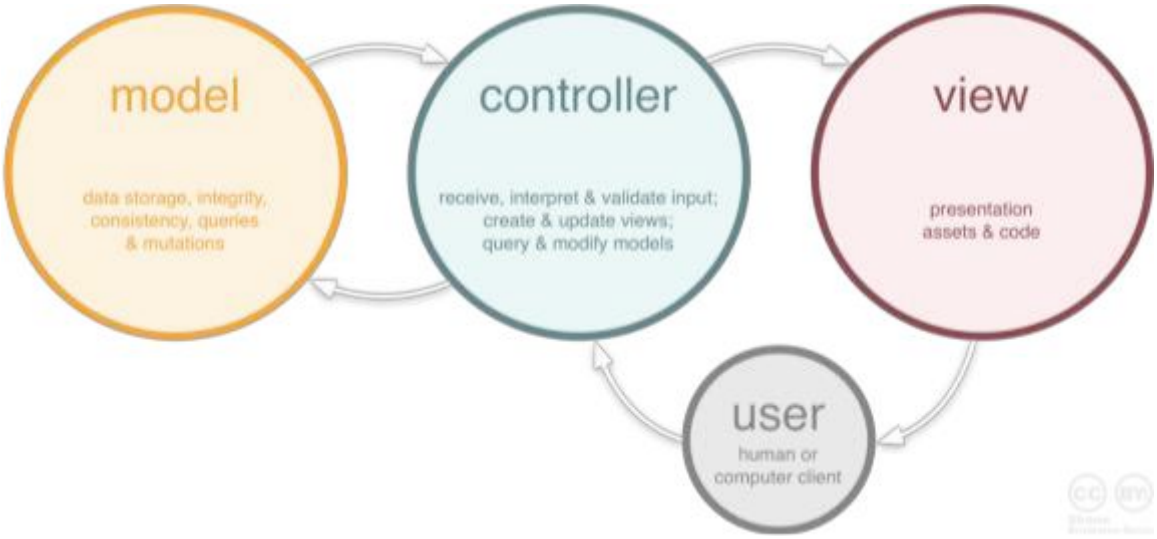

Figure 4: MVC Framework

Django uses a database, usually SQLite, to manage and store the session data, and Hue applications can use it as well for the models. For programmers, the essential thing to know is how the "urls.py" file maps between URLs and view functions. These view functions typically use their arguments (captured parameters) and their request object (the POST and GET parameters) to prepare dynamic content to be rendered done by using a template.

3 Making the HUE Application

3.1 Basic HUE Application

First of all, everything necessary that should be integrated in one HUE application should be installed. On CentOS (the OS used for this project) these were required: Oracle's JDK, ant, asciidoc, cyrus-sasl-devel, cyrus-sasl-gssapi, cyrus-sasl-plain, gcc, gcc-c++, krb5-devel, libffi-devel, libtidy (for unit tests only), libxml2-devel, libxslt-devel, make, mvn, mysql, mysql-devel, openldap-devel, python-devel, sqlite-devel, openssl-devel, gmp-devel.

After the environment preparation, the application should be created and installed with the create_desktop_app command. Once everything is set up, it is possible to start developing the HUE web application itself. Considering that HUE application is a Django application in core, views and templates should be made first and the url.py file has to be modified. There are some minor differences in the code compared to the development of a standard Django application, especially because Mako language is used for templates.

The result of this is application which is running on http://localhost:8000/.

Figure 5: HUE Authentication

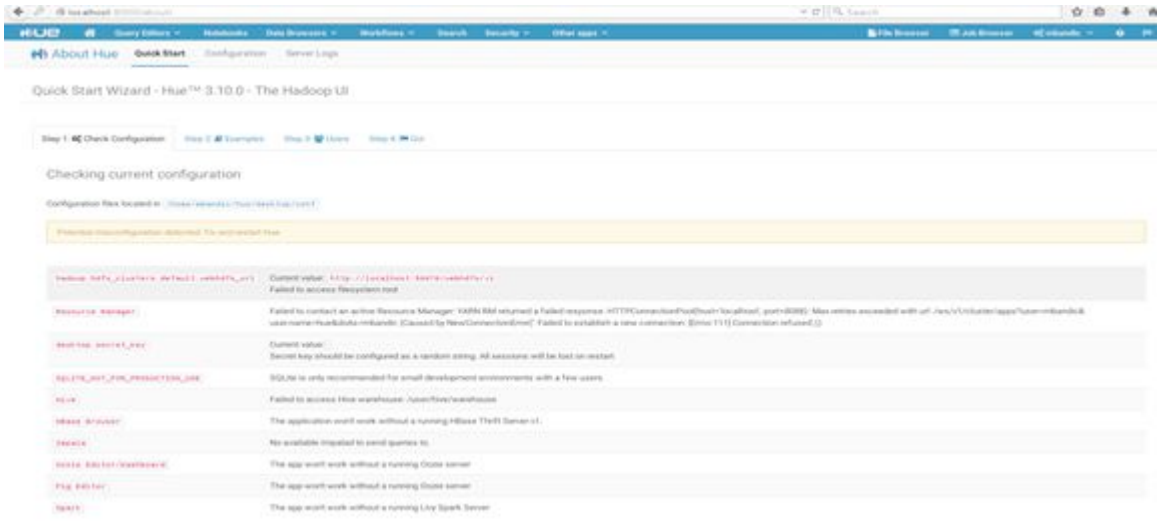

Figure 6: Basic HUE interface

Figure 7: "The Hello World" HUE application

Usually the models are also necessary, but in this case instead of making the database through models, the application is connected with the HLoader REST API which uses PostgreSQL as backend.

3.2 HLoader

HLoader is a framework which is built around Apache Sqoop for data ingestion jobs between RDBMS and Hadoop Distributed File System (HDFS). It has been developed at CERN to facilitate and automate the data ingestion from relational databases from a wide range of projects including some CERN experiments and critical systems of the CERN Accelerator Complex.

The goal of the HLoader project is to automate the process of data ingestion between RDBMS systems and Hadoop clusters by building a framework that is like a data pipeline, and interfaces with the systems via Apache Sqoop and other tools, applying all the necessary actions in order to deliver ready-to-read data on the Hadoop file system for high-end frameworks like Impala or Spark.

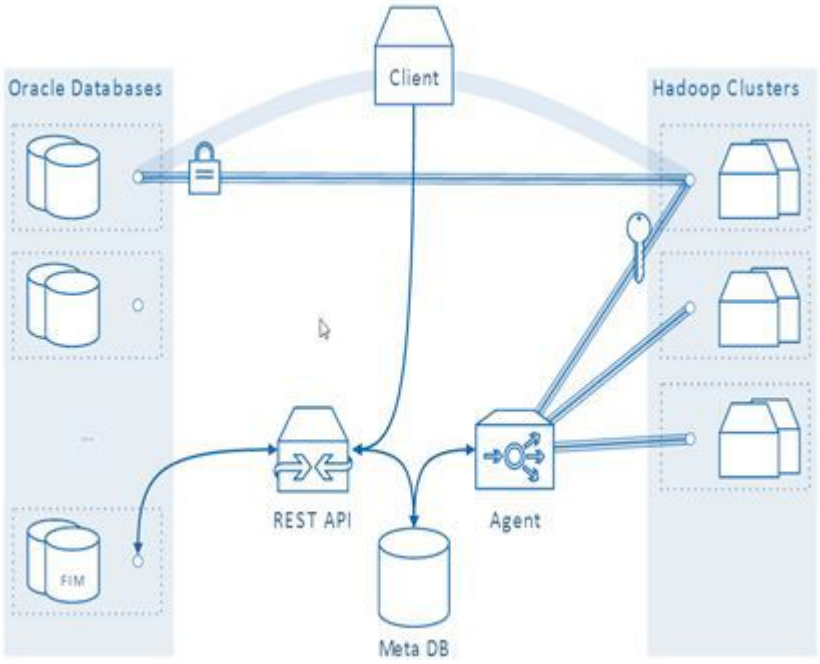

Figure 8: HLoader Framework

HLoader components include an agent, a scheduler, runners and a monitor. The agent is responsible for populating the meta-database with newly added jobs notifying the scheduler of the changes. When a new job is added to the metadata database, the user provides all the information required by Sqoop for the transfer. The transfer initiates a Sqoop Map Reduce job which imports data from the database to the target Hadoop cluster. The metadata database also stores all the information related to the transfers, including logs and execution status.

The users interact with the HLoader through a RESTful API, which retrieves some of the data in the metadata database to authenticate the user and allow them to perform basic CRUD operations.

3.3 REST API Connection

In this project, the web application is going to be connected with some different REST APIs to retrieve the information. To facilitate the development and testing, the connection with the REST has been first done using the command line and then code has been ported to the HUE web application. The same approach was taken for the implementation of the views, writing the custom functions and using them on the templates.

On the other side, a development HLoader REST API environment has to be set up. The procedure for this required first to download the HLoader project and install its dependencies using the python requirements.txt and the Oracle.rpm files. The next step is to configure the config.ini file (*AUTH_* properties for the authentication database and *POSTGRE_* properties for the meta database). The metadata database should be also set up using the scripts provided in the HLoader project.

Once everything is configured, the HLoader REST API can be started using the HLoader.py file. The REST API will be exposed to the user the metadata and will allow the submission of new jobs. The HLoader REST API runs by default on http://127.0.0.1:5000. The jobs submitted using the API will be executed using Oozie Workflows or Coordinators.

Figure 9: Part of HLoader's JSON

The following list describes the available methods in the HLoader REST API:

GET /headers

Returns Python environment variables and request headers.

GET /api/v1

The index page

GET /api/v1/clusters

Returns a JSON with an array of clusters, potentially filtered by an attribute value.

GET /api/v1/servers

Returns a JSON with an array of servers, potentially filtered by an attribute value.

GET /api/v1/schemas

Returns JSON with arrays of available and unavailable schemas given an owner username.

GET /api/v1/jobs

Returns a JSON with an array of jobs, potentially filtered by an attribute value.

POST /api/v1/jobs

Submits a job and returns a JSON containing its ID

DELETE /api/v1/jobs

Deletes a job given its ID and reports the status of the operation.

GET /api/v1/logs

Returns a JSON with an array of logs, potentially filtered by an attribute value.

GET /api/v1/transfers

Returns a JSON with an array of transfers, potentially filtered by an attribute value.

On Figure 10 we can see how the list of jobs and the details are shown. The functionality to delete a job is also possible using the delete button.

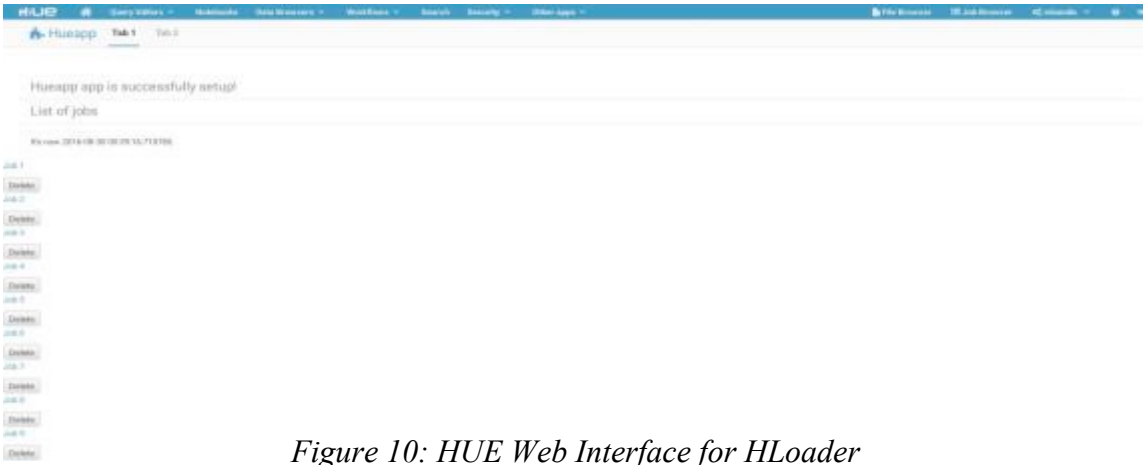

Figure 10: HUE Web Interface for HLoader

4 Future

The next steps of the project include to complete the web application design and its integration with the REST API – at the moment just the core set of operations have been integrated. The application should also be extended to support a larger number of ETL functionalities, as well as more data sources and data types. Once the application is fully functional and the core operations are completely supported and tested, additional integration tests have to be prepared to ensure the proper functioning with the CERN IT infrastructure. Finally, once the integration is ensured, the application has to be deployed in the production environment.

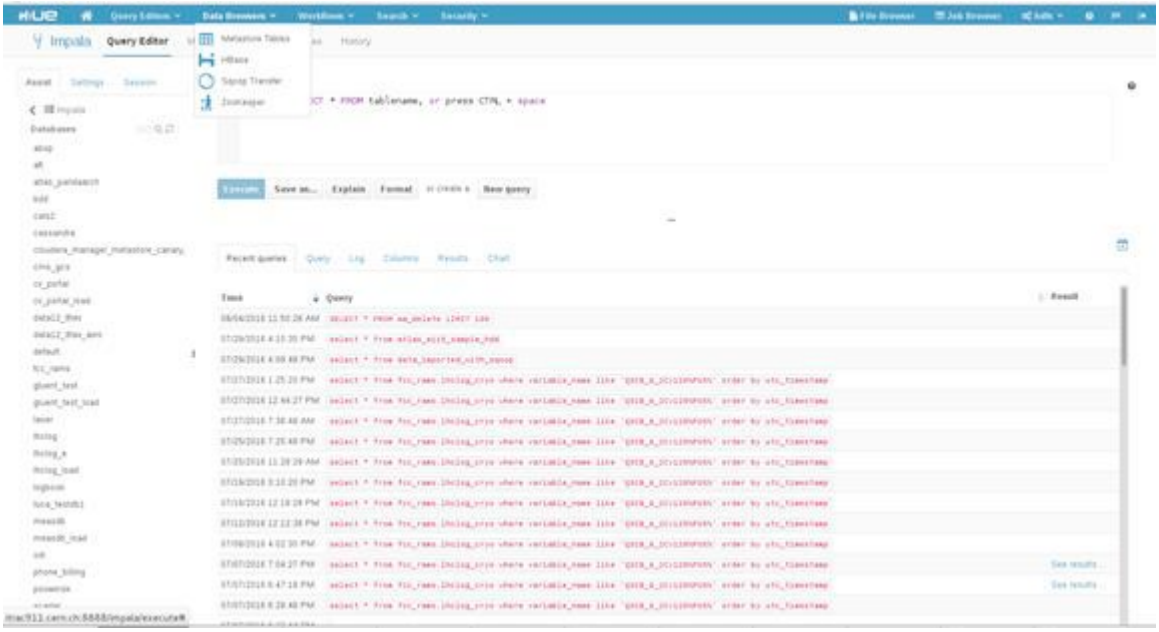

Figure 11: Example of Finished HUE Interface

5 Conclusion

The automation and simplification of the ETL process into the CERN Hadoop infrastructure plays a key role for the CERN users and the service administrators. With this arguments, a web based interface makes possible to both to control and create new ETL process in a fast and comfortable environment.

The integration of the application into the HUE framework provides many built-in functionalities for the big data ingestion process. On the contrary, the development can be challenging at the beginning mainly because of the lack of technical documentation. A large amount of time was spent in the first stages of this project due to this issue.

Nevertheless, once the developers are familiar with the technology, the framework can greatly simplify and improve the implementation of interfaces to the big data ingestion process, especially in complex environments with very large amount of data like the CERN.

Acknowledgements

First of all, I want to thank to my supervisors Antonio and Manuel for the guidance and support during the whole summer. They helped me to face and fix all of the issues and thanks to them I can say that learned a lot through developing this project.

I would also like to thank the best group in CERN – the IT-DB group for providing us students a really nice working environment during the whole stay.

In the end I would like to thank CERN for giving me opportunity to have the best summer with all of the amazing people I met and places I had opportunity to see.

It was a great honor for me to work in CERN for two months and I hope I will get a chance to return back some time in the future as well.

References

thenewboston.com - Django Tutorial for Beginners www.coursera.com

– Hadoop Platform and Application Framework

http://cloudera.github.io/hue/docs-2.5.0/sdk/sdk.html#fast-guide-to-creating-a-new-hue-application

https://github.com/cerndb/hloader

Achari, Shiva "*Hadoop Essentials delve into the key concepts of Hadoop and get a thorough understanding of the Hadoop ecosystem*".Birmingham :Packt Publ., 2015.