DIGIT.B4 – Big Data PoC

GROW – Transpositions

D04.01.Information System

everis Spain S.L.U

Table of contents

Figures

1 INTRODUCTION

1.1 Context of the project

The objective of this proof of concept showcasing the use of big data in the procurement domain, in cooperation with DG GROW, is to prove the usefulness and policy benefit that big data can bring.

This proof of concept shall also demonstrate the use of natural language analysis techniques to check the compliance of the transpositions sent by the European Member States related to EU directives. In the context of the PoC, one directive and its respective national transpositions will be analysed, with the objective of supporting the manual checks currently done by European Commission staff.

1.2 Objective

The aim of this document is to describe all technologies used, the tool structure and how to deploy and run the tool.

2 TECHNOLOGIES USED

In this project, multiple free technologies have been used for the publication of the results. These technologies are the ones most frequently used in the data analysis and visualisation fields of knowledge and are widely supported by different software communities around the world.

2.1 Python

Python is an interpreted, object-oriented, high-level programming language with dynamic semantic. Its high-level built-in data structures, combined with dynamic typing and dynamic binding, make it very attractive for rapid application development, as well as for use as a scripting or glue language to connect existing components together.

Its philosophy emphasises code readability and its syntax allows users to express complex concepts in fewer lines of code than in other languages such as Java or C.

The Python interpreter and the extensive standard library are available in source or binary form free of charge for all major platforms, and can also be freely distributed.

These are the main reasons to choose this language as the base for the project.

In this project, we have used many Python libraries that are discussed below.

The data analysed in this project has been written in CSV format, therefore the CSV library was used as it implements classes to read tabular data in CSV format. Additionally, programmers can describe the CSV format understood by other applications or define their own special purpose CSV formats.

To save the analysed data for later viewing we used MongoDB, which saves data in JSON format. Therefore, a JSON module has been used to read and write data.

Also, the os module has been used to access operating system dependent functions, as well as the sys module, which provides variables and functions directly related to the interpreter.

2.2 Django

Django is a free and open source web application framework, designed for Python, that allows rapid development and prototyping. It takes care of the most common tasks performed by web developers so they can focus on writing the app instead of dealing with repetitive work.

In this project, Django has facilitated the use of a unique technology, because the development of data modelling has been done with Python programming language.

Django is the only framework that has the ability to generate admin panels on the fly depending on the database schema and table relations. That makes Django the most powerful framework today when it comes to sites where admin panels are widely used. This is the main advantage over other frameworks like PHP. Some of the benefits that Django offers over other frameworks include that the ability for the administrator to add new users with specific rights on the fly, as well as making user groups with rights to edit/delete content. It also has a simple but useful design (everything that any administrator will ever need is there: action history, adding/updating/deleting content,

uploading images/videos/files, etc.), and better database handling to provide easy data manipulation, migration, etc.

It supports the most popular web application servers (as Apache and Nginx) and databases (MySQL, PostgreSQL).

2.3 JavaScript

JavaScript is a programming language that is mainly used to create dynamic web pages.

Its simple language and easy integration are one of its main advantages, as well as its compatibility with most browsers.

In this project, it has been used to control the actions of visitors and define the tool behaviours when they occur.

2.4 HTML & CSS

HTML (Hypertext Markup Language) and CSS (Cascading Style Sheets) are two of the most popular core technologies used to build web pages. HTML is the language that describes the structure of web pages while CSS describes the presentation, covering the colours, layout and fonts. It allows the developer to adapt the presentation to different types of devices, such as large screens, small screens or printers. CSS is independent of HTML and can be used with any XML-based markup languages.

These are probably the most widely used web technologies, for the numerous benefits they have as well as broad support. HTML is an open technology, highly flexible and supported on almost every browser. CSS will automatically apply the specified styles to the elements desired. The web pages also load faster and, thanks to the simplified structure and location of CSS files, changing the style of an element is much easier and faster.

These technologies have been used in conjunction with Django for the website presentation layer.

Django also provides its own template language to work with HTML, making the work easier for developers as it makes the presentation of the application more dynamic.

2.5 Amazon EC2

One of the objectives of the project is to be as agile as possible in the creation of prototypes and versions of the application.

The Amazon EC2 (Elastic Compute Cloud) service provides a very fast and easy server deployment process, allowing the developers to ignore the details of preparing a specific machine or machines for the web application.

As a service, these web servers also provide great scalability and security, and the possibility of choosing the operating system that best suits the needs of the project.

2.6 Apache HTTP Server

The Apache HTTP Server is an open source HTTP web server for Unix platforms (BSD, GNU / Linux, etc.), Microsoft Windows, Macintosh and others, which implements the HTTP / 1.12 protocol and the notion of virtual sites.

Its long history of reliability and performance have resulted in a large amount of documentation, and it is very easy to get help with any trouble. In addition, it is free and there are no licensing fees or costs.

It is one of the most feature rich web servers available. There isn't much it can't do.

In order to put the project into production, Apache HTTP Server was used. The module “mod_wsgi” was used to do it. “mod_wsgi” is an Apache module which can host any Python WSGI application, including Django.

2.7 MongoDB

MongoDB is the most famous data base within NoSQL databases, highlighted for managing large volumes of data (big data).

MongoDB is a database oriented in documents; it stores the data in documents instead of records.

These documents are stored in BSON format, which is a binary representation JSON. BSON may represent, in a single entity, a construction that will require multiple tables to be displayed in a relational database. Also, this structure is analogous to Python dictionaries, which are very complex structures but very easy to drive.

In this PoC, all information related to the directives and transpositions was stored in JSON format in MongoDB.

The data is stored as horizontally scalable clusters, which is much easier than other relational databases.

Furthermore, it is not necessary to define a model schema as the paper defines the schema data and queries can be more dynamic.

3 PROJECT STRUCTURE

3.1 Project structure

Django offers a very intuitive structure for a project, separating each part of the project in different areas. The following sections describe the application structure that was developed using the Django framework.

3.1.1 **Application structure**

A Django project may consist of several apps, each one with its specific directory within the main project.

In this case, only one app has been developed: the app for the visualisation of the compliance and completeness checks performed on the transpositions of European Commission directives into national legislation. Thus, the directory structure is as follows:

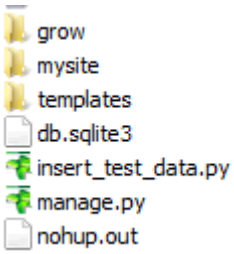

Figure 1 - Project structure

In Figure 1 the structure of the application is shown. The folders and most important files are the following:

- Grow: contains the app;
- Mysite: belongs to the principal project;
- manage.py: Django script to run several tasks not related with the development.

The grow folder contains the principal app and is structured as shown in the following image:

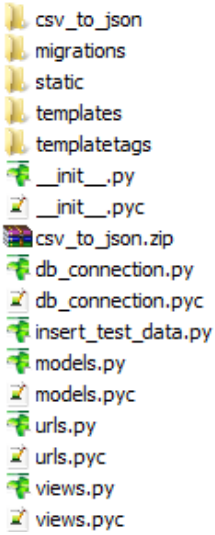

Figure 2 - App structure

3.1.2 Application logic

The application logic can be found in the root directory for the app. This root directory contains the following files:

- "db.connections.py" file: the connection to the database and all necessary queries are defined;
- "models.py" file: all models related to the project are defined;
- "properties.py" file: the static variables used in the project are defined;
- "urls.py" file: the URLs for each web page are defined;
- "views.py" file: the functionality of each web page is implemented.

3.1.3 Application templates

The web application uses HTML templates to show the results and the information gathered from the analysis. These templates are located in the templates directory, and are linked to the views.

3.1.4 Application static files

Web applications usually use some kind of static files for the presentation layer. In this case, the static folder contains:

- Images of the application (logos, icons, etc.);
- JavaScript functions and events: static script files to control the actions of the user and define the tool behaviours when they occur;
- CSS files: the CSS style sheets are also located in this folder, as they won't dynamically change in the application.

In addition, the csv_to_json folder contains the necessary processes to convert a CSV file into a JSON file because the analysed information is generated in a CSV format. Then, this information is stored in MongoDB, which requires the information in JSON format.

4 MAJOR TASKS TO RUN THE TOOL

The major tasks in the migration of the application are explained in this section.

4.1 Installing Django framework

The Django framework can be installed by following the steps on the official website, which are very straightforward:

https://docs.djangoproject.com/en/1.9/intro/install/.

The easiest way is to install Django with an official release:

https://docs.djangoproject.com/en/1.9/topics/install/#installing-official-release

Apache 2 should be installed before Django.

To quickly install Apache, the official documentation for mod_wsgi has a very straightforward guide:

https://modwsgi.readthedocs.org/en/develop/user-guides/quick-installation-guide.html

Django does not need any special configuration to run the PoC application.

4.2 Deploying the application

Once Django is installed, the PyMongo module must be installed to run the project. To do this, the user must write the command in the Linux console:

```
> pip install pymongo
```

Then, the source code must be imported into a Linux console.

The tool is composed by one folder to import:

The next steps describe how to import the tool:

1) Open the “linux” console
2) Copy the folder in the Apache directory `/var/www/`
3) Change its ownership with the following command:

```
> chown –R www-data:www-data /var/www/grow_project
```

Then, an Apache configuration file must be created in the directory /etc/apache2/sites-available as follows:

1) Execute the following command

```
> touch /etc/apache2/sites-available/grow.conf
```

2) Disable the default Apache setting with the following command in `/etc/apache2/sites-available/` directory

```
> a2dissite 000-default.conf
>service apache2 reload
```

3) Create a new configuration file

```
> nano grow.conf
```

```
<VirtualHost *:80>
        WSGIScriptAlias / /var/www/grow_project/mysite/wsgi.py
        Alias /static/ /var/www/grow_project/grow/static/
        ServerName [Ip machine]
        <Directory "/var/www/grow_project/">
                Order deny,allow
                Allow from all
        </Directory>
        <Directory "/var/www/grow_project/grow/static/">
                Order deny,allow
                Allow from all
        </Directory>
</VirtualHost>
```

4) Activate the new configuration

```
> a2ensite grow.conf
> service apache2 reload
```

Finally, the user must give permission to the project .To do this, he/she must execute the following command in the `/var/` directory.

```
> chown -R www-data:www-data www
> chown root:root www
> service apache2 restart
```

In addition, the data project is stored on MongoDB as we mentioned above, so MongoDB must be installed.

There is a guide on how to quickly install MongoDB:

https://docs.mongodb.org/manual/tutorial/install-mongodb-on-ubuntu/

Then, the bind_ip field of the mongodb.conf must be changed to the IP machine where the project is, by executing the following command:

```
> service mongodb restart
```

Also the “IP_MONGODB” in the properties.py file in the projects files must be changed to the IP machine where MongoDB was installed.

The application should now be up and running on http://[IP_address]/grow/home

Finally, each directive must be a document in JSON format in order to insert the data into the database as shown below:

```
{
"directive":{
            "nam_directive"
            "des_directive"
            "url"
               }
"spatial":[
                              {
                              "country"
                              "language"
                              "compliance"
                              }
                              {
                              "country"
                              "language"
                              "compliance"
                              }
                              …
               ]

“name_country_language”:{
"articles" :[
                        {
                              "serial_article"
                              "des_article"
                              "compliance"
                              "status"
                              “keywords”
                        }
```

```
                {
                        "serial_article"
                        "des_article"
                        "compliance"
                        "status"
                        “keywords”
                }
                …
        ]
"paragraphs":[
                        {
                        "des_paragraph"
                        "compliance"
                        }
                        {
                        "des_paragraph"
                        "compliance"
                        }
                        …
                ]
        }
}
```

Scheme			Type	Description
directive			Dictionary	Dictionary representing a particular policy
	nam_directive		String	Directive short name
	des_directive		String	Complete directive name
	url		String	Original directive´s url
spatial			List	List of transpositions
	country		String	Transposition country
	language		String	Transposition language
	compliance		Float	Percent of similarity of the transposed directive with respect to the original
	articles		List	List of articles translated from the directive
		serial_article	String	Article name
		des_article	String	Article name
		compliance	Float	Percent of similarity of transposition with respect to the original article
		status	String	Acordance indicator
		keywords	List	Keywords for language transposed directive
	paragraphs		List	List transposed paragraphs
		des_paragraph	String	Paragraph content
		compliance	Dictionary	Dictionary of similarity percentages for each article of the directive.

The following command must be executed to insert the data into the database:

```
> mongoimport --db grow --collection directives --file name_file.json --jsonArray
```

4.3 Post-implementation verifications

The first time a user accesses the application via URL to the Home section, the following screen should appear:

Figure 3 - Home section

All storage directives should appear as in the previous figure, where the “DIRECTIVE 2011/7/UE” is shown. In addition, if the directive is clicked, all analysed transpositions will appear as in the following figure:

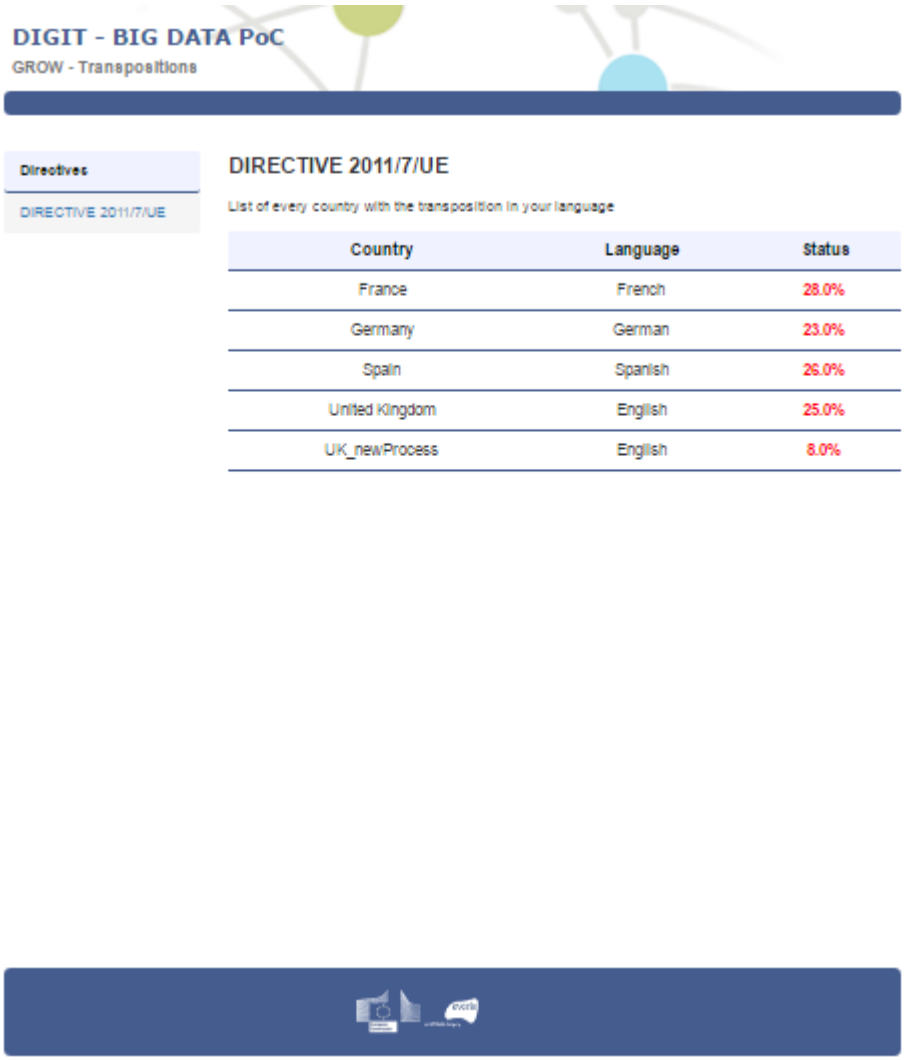

Figure 4 - Analysed transpositions