Studies in Informatics and Control, 26(2) 249-258, June 2017 ISSN: 1220-1766 eISSN: 1841-429X 249
https://doi.org/10.24846/v26i2y201713

Cloud Computing Technology to Assist Government in Decision Making Process

Eugen Ștefan Dorel COJOACĂ[1], Mirona Ana-Maria POPESCU[2], Gabriel Cristian AMBĂRUȘ[3]

[1] Ministry of Communications and Information Society, Liberty Avenue, Number 14 Bucharest, 050706, Romania, eugen.cojoaca@gmail.com
[2] Politehnica University of Bucharest, Splaiul Independenței 313, Bucharest, 050706, Romania, mirona.popescu15@gmail.com
[3] Politehnica University of Bucharest, Splaiul Independenței 313, Bucharest, 050706, Romania, gabi_ambarus@yahoo.com

Abstract: In order to streamline government's internal decision making process that may be conducive to obtaining non-refundable European funds, this paper proposes new and intelligent methods for reporting, collecting and processing data in an updated manner, taken from websites of Eligible Competent Authorities to conduct such funds. Retrieving data will be made by means of intelligent mobile agents residing on the contracting public authority servers. Such data are processed in the cloud to generate reports or graphs, with the aim of measuring the EU FUNDS absorption rate at a certain time with a view to finding the best decisions that can increase EU funds absorption rate at national level and in the same time provide, ahead of time, the necessary budgetary funds for co-financing the European projects that have been undertaken.

Keywords: Cloud computing, intelligent agents, EIF, APPP.

1. Introduction

The European Union (hereinafter "EU") politics about digital transformation of government and the action plans in the field, aim to accelerate eGoverment in Europe through existing management interoperability of electronic services, which should be highly reliable and available and with a high utilization rate. Information provided to citizens or civil society (to provide transparency in spending public funds) through portal sites of each Competent Authority (hereinafter "CA") are the following: the total expenditure of central government staff, spending on goods and services, costs on other transfers, expenditure on projects, Non-Reimbursable European Funds (hereinafter "FEN") capital expenditures, research costs on the development and implementation of intelligent methods to get data from centralized reports used in the decision making process that can demonstrate the effectiveness of the programs.

An updated status and processing of the relevant data has become a necessity of e-Government. Most institutions keep their data in Microsoft Office Excel (.xls) documents. However, gathering the information in databases would enable data processing and interpretation.

Currently, there is no standardized reporting format (template), therefore it was created a first model based on all the reviews received by the Ministry of Communications and Information Society of Romania (MCSI), from the Ministry of Public Finance (MFP) of Romania. Any reporting to MFP takes at least two weeks and a minimum of three people involved, which leads to a very high cost unless a reporting method is available. The reporting requests received by MCSI are in a non-electronic format and most often they contain an attachment represented by a table head as a reporting template. The same reporting is required at the annual control by the Romanian Court of Accounts, and they have a different template. Introducing a reporting standard avoids human errors and promptly points out all reporting or accounting errors. This will prevent from making further errors and further corrections.

The present paper proposes a concrete solution through a user-friendly platform that would gather information in real time through intelligent agents from the institutions websites. A priority represents the focus on automating existing processes within the CA to increase the effectiveness of Public Administration such as, for example, the intention of the Ministry of Regional Development and Public Administration of Romania to provide a general system for the Public Authority in which all processes are automated.

In order to acquire fast and comprehensive reports reflecting the implementation status of EU funded projects at a proper time, new methods of reporting (such as, an Annual Public Procurement Plan, hereinafter "APPP"), retrieving and processing data using Cloud properties are used. Given the fact that the relevant information is shared with relevant stakeholders, intelligent agents were used for extracting and transmitting it in a cloud (such as IBM BLUEMIX) for a fast processing and for providing the results of this processing to as many beneficiaries as necessary.

The resulted reports or graphs can be generated for a certain period (e.g., month, year, etc.).

2. State of the Art

There are many fields in which intelligent agents are used for development purposes and this is still a topic of major interest, as researchers continue to improve these intelligent agents. However, at the present moment, in e-Government there is not yet available a software that uses intelligent agents to collect data from public institutions and store them on Cloud so that they can be analysed through a user-friendly webpage interface.

According to article [7], past years have witnessed the development in the field of mobile learning, fostered by the continuous progress in mobile computing and wireless technology. The purpose of the paper studied is to show the model and simulate the ambient mobile system based on intelligent agents. The mobile agent is based on the distributed client server. The article studied proposes a mobile intelligent agent based architecture for the MLearning that aims to facilitate the teacher and student relationship and an area of development which became a principal tool for our education system. The article introduces the scope and the background emphasizes m-learning as the next generation of e-learning, introduces the AMMAS (Ambient Mobile Multi-Agents System) model for the M-Learning and the system implementation.

In article [8] wireless sensor networks have embedded the concept of mobile agent to decrease energy consumption and effective data gathering. Typically, in mobile agent based data gathering, one of the steps is to find out the most efficient itinerary planning for the mobile agent. Using multi-agent itinerary planning solves the drawbacks of single-agent itinerary planning. In spite of the advantages of multi-agent itinerary planning, the optimal solution of distributed mobile agents is still a difficult issue. In the article referred to, the authors discussed the algorithms that have been identified in the literature to address the above issues. Their article shows that most of the algorithms use one parameter to compute the optimal number of mobile agents in multi-agent itinerary planning.

Authors in article [5] state that mobile robots are regarded as a solution to overcome the current limitations of fixed automation systems, particularly in the area of large structure assembly. The main challenges for traditional automation systems in large structure assembly are the transportation of products and the adaptation of manufacturing processes. The paper discussed proposes a Multi-Agent System (MAS) approach, applied to self-organise mobile robots in large structure assembly, based on fixed-priority emptive scheduling with a blackboard agent as a central information source.

The article [9] presents a bio-inspired mobile agent-based coalition formation system for recruiting modular robots into different teams, through mathematical model for the coalition formation, through a mobile agent-based system aiming at bridging the gap between theory and practice in robot's coalition formation. Unlike the centralized system, each robot makes its own decision based on the ant-colony algorithm. Moreover, the described system has a high flexibility in integrating various algorithms in order to attain different requirements of robot recruitment. System architecture and implementation details are presented in the article referred to, as well as a coalition formation example to illustrate the properties of the system.

In article [6], mobile agent technology builds a innovative computing paradigm in which a program, in the form of a software agent, can suspend its execution on a host computer, transfer to a different agent-enabled host on the network, and resume execution on the other host. Recent mobile agents are able to categorize in different ways of distributed equipped embedded intelligence agents systems. This information provides details of the variety of threats affecting the mobile agent systems as developers of mobile agent facing

security problems based on their applications. The business information agent system is built based on mobile agent technology, sends out agents to different hosts in an electronic marketplace, and the agents collect and report information such as prices and availabilities about products as user-generated. Security is a main issue of mobile agent systems, particularly when cash transactions are concerned.

3. Data Model

Currently, the reporting process carried out by the CA is made in compliance with certain criteria, but the published data are not useful for the automatic processes. If the data is published in an appropriate format, intelligent mobile agents will be able to extract more relevant information from these data.

One example can be the case when it is mandatory to establish the efficiency of specific CA in European Fund absorption. The solution proposed assures publishing of statistical data in a format better suited to stakeholders and this implies an automated process for efficiency.

Having in view the current Romanian laws, an example of reporting (such as APPP) can betaken from public sources (website from a CA) [12].

This reporting manner has some disadvantages:

Firstly, the reports are made on a daily basis, so it is very hard to follow the expenses on a period longer than a few months. Usually a project carried out by a CA covers minimum five years due to its complexity.

Secondly, the expenses are not presented in a cumulative manner and for this reason the time necessary for each person who wants to know the total expenses for a single project, will increase proportionally with the implementation period of the project.

Thirdly, if the CA has more than two projects under implementation, the complexity of following expenses for these projects will be a difficult task for a human operator.

Fourthly, it is impossible for a citizen to know the expenses for all European projects from all CA at a specific moment in time.

Finally, it is impossible to apply automated methods for processing all these data because is hard to distinguish which expenses are related to current activities of CA and which ones are made for European Projects implementation, etc.

For all the reasons mentioned above, we have proposed a new reporting template, presented in Figure 1.

			The amount paid			
		Total	Trim I	Trim II	Trim III	Trim IV
Staff costs	State budget	13.729.000,00	3.464.000,00	3.458.000,00	3.446.000,00	3.361.000,00
	EIF	100.000,00	25.000,00	25.000,00	25.000,00	25.000,00
	Total	13.829.000,00	3.489.000,00	3.483.000,00	3.471.000,00	3.386.000,00
Expenses for goods and services	Maintenance	745.833,33	186.458,33	186.458,33	186.458,33	186.458,33
	Goods	216.666,67	54.166,66	54.166,66	54.166,66	54.166,66
	Services	1.687.868,42	421.967,10	421.967,10	421.967,10	421.967,10
	Total employee	2.650.368,42	662.592,09	662.592,09	662.592,09	662.592,09
	State budget	3.150.000,00	750.000,00	800.000,00	800.000,00	800.000,00
	Economy State budget	499.631,58	87.407,91	137.407,91	137.407,91	137.407,91
Expenses for other transfers		1.300.000,00	1.300.000,00	0,00	0,00	0,00
EIF	Sustainability Cod SMIS x	360.000,00	90.000,00	90.000,00	90.000,00	90.000,00
	Sustainability Cod SMIS y	300.000,00	75.000,00	75.000,00	75.000,00	75.000,00
	Sustainability Cod SMIS z	320.000,00	80.000,00	80.000,00	80.000,00	80.000,00
	Sustainability Cod SMIS t	4.000.000,00	1.000.000,00	1.000.000,00	1.000.000,00	1.000.000,00
	Total employee	4.980.000,00	1.245.000,00	1.245.000,00	1.245.000,00	1.245.000,00
	Amounts allocated	6.893.000,00	1.250.000,00	3.143.000,00	1.250.000,00	1.250.000,00
	Amounts Neasorbit	1.913.000,00	5.000,00	1.898.000,00	5.000,00	5.000,00
Capital expenses	Repairs	34.000,00	0,00	17.000,00	17.000,00	0,00
	Investment	1.500.000,00	0,00	0,00	1.000.000,00	500.000,00
	Study / soft	51.578.000,00	41.394.000,00	105.000,00	4.917.000,00	5.162.000,00
	Total	115.959.000,00	104.241.000,00	122.000,00	5.934.000,00	5.662.000,00
Current expenses		27.485.000,00	6.942.000,00	7.644.000,00	5.888.000,00	7.011.000,00
Research expenses		1.892.000,00	0,00	0,00	347.000,00	1.545.000,00
Grand total State budget		162.215.000,00	115.397.000,00	12.024.000,00	16.415.000,00	18.379.000,00
TITLE X Projects financed from external grants financial framework 2014-2020	Programs of the European Regional Development Fund (ERDF)	5.000.000,00	1.250.000,00	1.250.000,00	1.250.000,00	1.250.000,00
	Programs of the European Social Fund (ESF)	505.000,00	0,00	505.000,00	0,00	0,00
	Grand total EIF	6.893.000,00	1.250.000,00	1.755.000,00	1.250.000,00	1.250.000,00
Grand total		174.613.000,00	117.897.000,00	15.534.000,00	18.915.000,00	20.879.000,00

Figure 1. The innovative reporting structure

This innovative model has the following advantages:

Firstly, is taken into consideration only budget chapters (e.g. the chapter regarding personnel expenses, the expenses from the state budget) to be presented separately.

Secondly, the expenses on every European Project will be presented in a detailed manner after the Structural Information Instruments Management System (hereinafter "SMIS") code, showing what has been paid in the current month and whether the reimbursement request was submitted.

Thirdly, the expenses made with feasibility studies will be better emphasized, this may represent an important factor as many feasibility studies could remain unused.

The expenses for goods and services and the expenses for maintenance of CA will be granted separate procedures. The model takes into consideration some prerequisite conditions such as:

- all the reported data must be found on a website maintained by the contracting authority;
- all the reported data should be published in a common format (for example, .xls);
- all the reported data should be published in a common template for all competent authorities;
- all data reporting should be expressed in the same unit of measurement.

4. Intelligent Agents and IBM Bluemix

4.1. Intelligent Agents

Hermans defined intelligent agents as: "An agent is a software that performs a given task using information extracted from its environment and acts in a manner appropriate to complete the task(s) successfully. The software should be able to adapt itself based on changes occurring in its execution environment, so a change in the execution environment will not influence the desired result. " (Hermans, 1997, p. 14) [2].

There are currently a variety of areas in which mobile agents can be used as they represent a growing interest in distributed systems and the focus is on the opportunity to have a higher mobility as a code. Mobile agents can move between certain sites on the Internet to collect data and execute the instructions designed by them.

Mobile agents generally have a heterogeneous character so by using programming languages such as Java or Python, they can run on different hardware configurations, which is an advantage because the government infrastructure is not homogeneous, but rather mixed.

One of the goals is to create intelligent mobile agents to get the data automatically from the competent authority's website, when the APPPs are introduced or when the existing ones are updated.

Due to their asynchronous and autonomous execution, the intelligent agents will connect to servers to extract data. The agents have the ability to work independently and to reconnect afterwards to the server to send the collected data.

In the carried-out developments was used a task-based approach of the intelligent agent because its purpose was known and consequently it could be predefined. To successfully perform tasks, the environment and the prospective changes must be taken into consideration.

4.2. IBM Bluemix

Bluemix is an implementation of the open cloud architecture provided by IBM, which is based on Cloud Foundry, PaaS type (Platform as a Service) and allows a user to develop and make cloud management applications in a shorter time. Being based on Cloud Foundry, Bluemix fits into a growing ecosystem frameworks and services. Besides, it offers the ability to create and view applications / services and the ability to monitor the developed application resources.

Due to its open source roots, Cloud Foundry is not a specific vendor and does not limit the use of proprietary software or required cloud infrastructure. Cloud Foundry abstracts basic infrastructure necessary to run a cloud, allowing the user to focus on the work to develop cloud applications. The platform is offering a wide range of options through its use.

The ability to provide options is due to build packages and an easy way to use the frameworks at runtime. Build packages can be

provided by the open source community and constructed in a personalized way.

Bluemix Dashboard is user-friendly to organize area spaces, users access and a range of predefined services that can be used within applications. Some of them belong to IBM vendors, others to third parties. The major advantage of this approach is to minimize the time spent developing an application. By using Bluemix there can be created instances of the necessary related services, without the need for additional configuration and allowing easy integration.

For developers, Bluemix optimizes the time spent for creating cloud applications. There is no longer a concern to install software or the need for images of virtual machines. This brings the benefit of reducing the number of hours designated for installation, configuration and troubleshooting. It streamlines the time spent for such rapid innovation.

The project implementation involved setting up a DevOps Services to run using Bluemix putting into production, the project being associated with an organization (org) and allocating space in the Cloud. DevOps services are used to recommend the practice of making an application plan, plan comprising the steps of development, test, and deploy the application's operation.

5. Implementation Model

5.1. Raspberry PI

Raspberry Pi has the technological performance of a smartphone or a notebook and has the dimensions of a HDD. It is a smaller computer that runs on an operating system based on Linux. In this project, a Raspberry Pi 2 and Raspberry Pi 3 were used.

The hardware configuration of a Raspberry Pi 2 is 900 MHz ARM Cortex A7 quad core CPU and 1GB of RAM. For data storage, it uses a microSD card with a minimum capacity of 4 GB. It has I/O devices as a HDMI port and 4 USB ports that can be used for external input devices such as keyboard, mouse, Wi-Fi module, and a LAN port.

Unlike the Raspberry PI 2, version 3 comes with a Wi-Fi module and Bluetooth Low Energy integrated on the device.

The application needs Raspberry Pi devices to host servers that simulate Romanian ministries. The data will be extracted from APPP documents by an intelligent agent.

5.2. Architecture deployment environment

Two types of raspberry devices, a router and a switch were used for the environment development (Figure 2). There were used a Raspberry PI 2 and a Raspberry PI 3. On each of them it was installed an operating system image with a dedicated version for each device.

The router is configured to allocate two static IPs as follows: 10.0.1.2 to version 2 and 10.0.1.3 to version 3 of raspberry. The first IP 10.0.1.2 has 8080 ports and 60022 and the other one which is assigned to Raspberry PI 3 has the ports 8081 and 60023.

Ports 8080 and 8081 allow the user to access the graphical interface where servers are hosting the APPP documents within a web browser through URLs mironap.go.ro:8080 and mironap.go.ro:8081.

60022 ports and 60023 are intended for the SSH connection and give the access to both devices raspberry to modify the configurations of these and to administrate the system.

Based on the MAC address of each raspberry device there is assigned a static IP from the range of the private addresses of the router so that when it attempts to connect to the Internet, the user does not receive a dynamically IP which can be different every time and it cannot be identified outside of the local LAN. The developer and users are connecting using a domain name, in this case mironap.go.ro instead of a public IP.

Due to changing of the public IP address to a certain time, it is used DDNS (Dynamic Domain Name System). DDNS maps in real time public IP domain name assigned by the developer. Thus, when the developer or the application users want to connect to their servers hosted by raspberry from a web browser by domain name, they will access the correct public IP even if this one has changed.

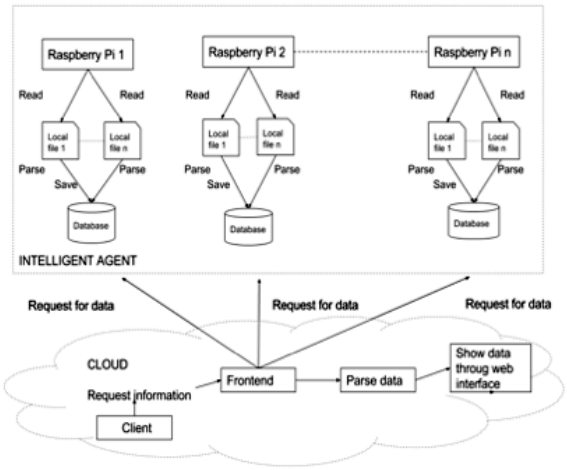

Figure 2. Development environment

5.3. Implementation (Stage one)

The two Raspberry PI are connected to a git repository. Thus, changes to the source code are instantly transmitted to them, always being up to date with the latest version of the software. For this to be possible it was used continuous integration (CI). This is a practice in which the isolated changes are automatically tested and reported when added to an existing project. Due to these reports, it can be easily and quickly followed any change and corrected when conflicts arise [1].

Raspberry PI simulates Romanian ministries from which the APPP documents are extracted by an intelligent agent. Each server (ministry) has the possibility to add new documents or to replace the existing ones, if the data contained changes along the way. Once a new APPP is loaded, it appears in the website interface hosted by IBM Bluemix. When a document is inserted it is also selected a date to match to it. Depending on this information, the agent knows how to extract and bring them into the interface used by a person to generate graphs and tables.

The server that hosts APPP and simulates ministries of Romania, was developed using the web framework Django open source, written in Python. The major advantage of using this framework consists of its performance and optimizations brought by Python [11], [3].

SQLite has been chosen as the programs can read and write directly to the database on disk without the existence of an intermediate process. The main advantage in using SQLite is the easy and quick access to the information stored in the database that requires no configuration from an administrator. It is also used for projects that do not have a broad scale and high volumes of data such as the present one [4].

The authentication on each server's platform (ministry) is made in the developed application based on a set of credentials, given by an administrator who also grants the privileges for the users.

The APPP documents are stored into a table designated to them. This helps the intelligent agent to identify them within each server. The agent identifies them by the date they were uploaded to retrieve them and to deposit them in the web application for users to handle the data.

APPP documents are maintained on the two servers that can be accessed through the following URLs (http://mironap.go.ro:8080 and http://mironap.go.ro:8081). If a request is sent to these servers from the application that resides on Bluemix on / get Years, it is returned a response as a list of years which shows how many documents are on that URL and the year of each one of them.

A user wishing to obtain data within the APPP existing on the web application makes a request to both servers and so it starts the search process of the document's year in the list of years. When the APPP is located, is sent back a response of type JSON that contains all the data within it.

It was used JSON because of its fast and an optimum modality to parse the information that is extracted and interpreted using graphs. The visual mode helps users to understand better how the funds were allocated and to track changes over time.

5.4. IBM Bluemix Application

Bluemix-hosted application can be accessed by a user who has access through an account to a web browser and an Internet connection. Access accounts are predetermined by the application administrator and there is a limited number of them. Thus, the work of people who want to use the application and have the corresponding rights can be more effectively monitored.

The application URL which is entered in the web browser is http://paapflask.mybluemix.net.

The user, once authenticated, is redirected to the main page of the application which have an easy and intuitive menu.

The application developed on IBM Bluemix is written in Python Bluemix and contains two screens, as follows:

One of them has the selection of the year. To display the years in the top menu, are interrogated the two servers of Raspberry PI, which respond with the years of the APPP documents they hold, following to be concatenated by the application.

The other one displays the data from the chosen APPP document and the graphics generated based on data extracted from these documents which have the .csv extension. Once the user has determined the year, the application server detects on what server the document resides and sends a request to obtain the data from the document.

The user can choose an APPP from the desired year through the menu on the left side of the screen, which lists the years. He runs a request to servers (ministries) represented by the Raspberry PI, to start the process of identifying on which of them the document resides. The answer is sent back within a few seconds and the user is redirected to another path of application where it can be viewed the APPP's contents as a table and the corresponding graphs. The user also has the option to choose in which form to be displayed the data as a chart (i.e line, pie, bar) through a dropdown menu.

After extracting the data from JSON, they are displayed to the user as a table like in Figure 2.1, with the associated graphs according to the selection.

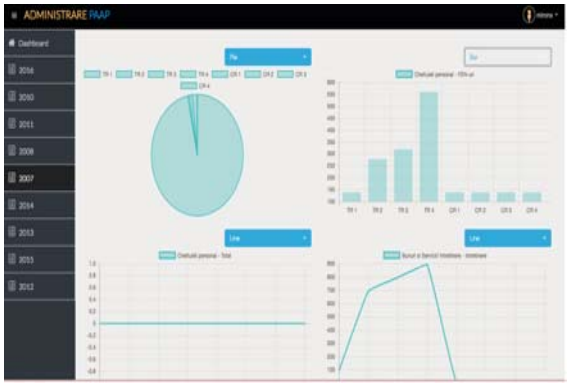

Figure 2.1. Obtained results

5.5. Backend

Flask was the appropriate framework chosen for developing the backend part of the application. This option was chosen because Bluemix does not support Django. Flask is a micro web framework written in Python and based on the toolkit and engine Werkzeug Jinja2 template. It is called micro framework because it does not require the developer to use a specific set of tools or a specific library. It has no database layer, form validation, or any other component of libraries from existing third parties common functions. However, Flask supports extensions that can add features to applications as if they were directly implemented in Flask [10].

All site routes, except for the login one, use the GET method. For each URL hold by the application, a check is made if the user is logged into the site. Through this manner, it is prevented the Security Through Obscurity ("STO") [11].

The access to the Dashboard's application is allowed if the user has initiated a session, otherwise it is redirected to the login page.

Documents from the two servers Raspberry PI are stored in JSON1 and JSON2 variables, to be easily interpreted because of the format, as stated above. The year associated to each APPP is retained into a vector. The function returns the user a view of the dashboard template with a list of years and the number of documents that reside on each of the two ministries simulated.

To obtain a APPP a user must send a request by pushing a button which in the back of the application it is a method called GetAPPPForYear that accepts as parameter the year. Based on this parameter, the search is performed and a response is re-turned.

The year sent in the request must be identified in one of two JSON files. Json1 stores the years of documents from port 8080 which is corresponding to server Raspberry Pi 1 and JSON2 those related to port 8081 from Raspberry PI 2.

These years are concatenated into the vector years' array. It is returned the APPP's template, the year of the document identified, the port on which was found and the years' vector in case there will be other subsequent searches.

5.6. Frontend

Frontend was implemented using the template inheritance model because of its major advantage represented by the fact that the template base can be inherited by children and be overwritten only the parts that differ, depending on the path accessed by the user.

All CSS and JavaScript files needed to develop the website are contained in base.html. To facilitate the development of the interface it was used Bootstrap framework, which benefits from all the tools HTML5, CSS, JS, and helps the developer to design a dynamic application.

Besides this, the application framework brings to page elements such as menus, buttons that can be customized using CSS.

5.7. Implementing the Intelligent Agent

The intelligent agent is implemented in the html file which is dedicated to the APPPs. It works by using an ajax script and has the role of parsing JSON documents. Using AJAX leads to a local interpretation of the data on the computer where the application is accessed through a website on the Internet, being avoided in this manner the server load.As the intelligent agent reads the data within the JSON it places the information into a table having the csv extension of files.

5.8 Implementation (Stage two)

Considering the previous model, using almost the same technologies, it can be created another model that implements an intelligent agent but has a different approach.

For example, instead of having a complex mechanism that memorise the data, a simple device that parse a standard XML file could be used. To achieve this scenario, also a Raspberry Pi could be used. This device would host an intelligent agent and he would be responsible for only one ministry.

On the other hand, the user should have easy access on data and, for this, every RPI would run a simple local server on which he would send and receive useful information.

Still, this is not sufficient. There still must be a two-way exchange of information and, for this, the user would use a local application that has its own server on which he sends all the information that he needs to provide, in order to obtain the necessary data.

To make the two entities work together, some rules had to be provided. Having an architecture that is displayed in Figure 2.2, a model of communication based on messages and tags would be useful.

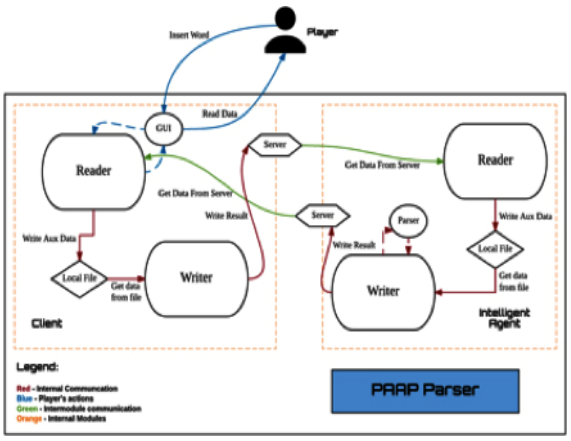

Figure 2.2. APPP Parser Architecture

For example, it is considered that the user needs information about the header of the table. It is a simple operation, but vital for the further implementation. Having the server, the user publishes the tag 1 (e.g. '*[1]*') on his local instance. On the other hand, the agent is notified when the status of the user changes and reads the data that had been sent to him.

Detecting the rule applied to the sent tag, the parser identifies the header of the file and sends it to its local server, having the tag 1' (e.g. '*[1']*') so that the user knows the information on the agent's server is the response to its request.

Still, to make the scenario functional, both the user and the agent were "taught" a dictionary that contains the rules which determine the operations. For the proposed model, the following rules or tags have been established:

- *"StandBy"* tag – user is not active, intelligent agent is standBy.
- *"1"* – table's header request.
- *"2"* – line's information type request.
- *"3"* – request that contains both the tag and a word for which the user requests data (line) retrieval.

• *"4"* – response to "3" type message, contains the words separated by commas.

• *"5"* – user requires a mathematical operation on line from the table (for example, the sum of some values).

• *"6"* – a response to the *"5"* type request.

However, knowing this is not sufficient. There are many ways of implementing this kind of architecture. The present example proposes a simple solution that works easily on a Raspberry Pi.

It is possible that one of the issues would be the communication between the devices. Therefore, it is necessary to use a programming language with functionalities regarding the creation of a local server (at least for simulation).

The most suitable solution for this issue is the use of Python programming language. Python has a plugin called webPy that creates a simple server which can be found at an address: *http://device_ip_address:8080*.

Also, another issue is the design of a basic graphic interface for the user, in order to give easy access to the application functionality. For solving this problem, Python programming language was used. It comes with a group of libraries called PyGame (used mainly in creating simple games) that provides all kinds of graphical functionality.

This is a simple example of the guideline that could be provided to have access to APPP information. An advantage of the simplicity is that the application is modular. There could be added multiple intelligent agents very easy, each one processing the information from a ministry.

6. Conclusions

This novel and innovative reporting method was developed because of the need to obtain a real statement of accounts of the public funding institutions involved in EU projects attracted by Romania. In this manner, the government will be able to know in every moment how many funds will be reimbursed, spent and how many funds will be necessary to complete the ongoing projects. This would increase the effectiveness of actions undertaken by PA and the information provided could benefit of a faster access and would be monitored in compliance with current requirements. The purpose is to create a general system of public administration for citizens, but also forauthorized persons in charge of European Projects and public procurement.

This software developed using IBM Cloud Bluemix, is designed to help the public administration system by collecting APPP documents from ministries' websites through an intelligent agent and to bring them locally to users. They can use the app on any device that has a browser and an Internet connection. The application interface is simple, easy and effective to provide reports and graphs within each APPP. Each user can choose the way the data will be displayed through a drop-down menu, so that the figures will not be difficult to interpret. Besides this facility, the application has another one which enables the user to search a public procurement procedure by its Common Procurement Vocabulary ("CPV") code.

The information is brought to the web application by an intelligent agent, which is notified when they are modified or a new APPP is loaded on any of the Romanian ministries. Thus, the application is constantly updating the data to be viewed by the users. A major benefit of this solution is given by the speed and scalability of the Cloud environment used. A second great advantage of the solution proposed is the new reporting model, which simplifies the data collection process. Also, it can offer citizens a higher transparency for public funds usage.

REFERENCES

1. Fowler, Martin, and Matthew Foemmel. "Continuous integration." Thought-Works http://www.thoughtworks.com/ContinuousIntegration.pdf (2006): 122.

2. Hermans, B. "Intelligent Software Agents on the Internet: An Inventory of Currently Offered Functionality in the Information Society and a Prediction of (Near) Future Developments," First Monday, Vol. 2, no. 3 (1997): 1–23.

3. Holovaty, Adrian, and Jacob Kaplan-Moss.*The definitive guide to Django: Web development done right*. Apress, 2009.

4. Owens, Mike, and Grant Allen. *SQLite*. Apress LP, 2010.

5. Ljasenko, Spartak, et al. "A Self-Organisation Model for Mobile Robots in Large Structure Assembly Using Multi-Agent Systems." Service Orientation in Holonic and Multi-Agent Manufacturing. Springer, Cham, 2017. 83-91

6. Malathy, M., S. Jasmine Smilee, and J. Niranjan Samuel. " Secure mobile agent in M-Commerce over internet." Emerging Trends in Engineering, Technology and Science (ICETETS), International Conference on. IEEE, 2016.

7. Mohamed, Shili, Moez Chebbi, and Santosh Kumar Behera & quot; AMMAS: Ambient Mobile Multi-Agents System: Simulation of the M-Learning" (2017).

8. Qadori, Huthiafa Q., et al. & quot; Multi-mobile agent itinerary planning algorithms for data gathering in wireless sensor networks: A review paper. & quot; International Journal of Distributed Sensor Networks 13.1 (2017): 1550147716684841.

9. Qian, Binsen, and Harry H. Cheng. " A mobile agent-based coalition formation system for multi-robot systems." Mechatronic and Embedded Systems and Applications (MESA), 2016 12th IEEE/ASME International Conference on. IEEE, 2016.

10. Perras, Joel. *Flask Blueprints*. Packt Publishing Ltd, 2015.

11. Van Rossum, Guido, and Fred L. Drake. The python language reference manual. Network Theory Ltd., 2011.

12. http://www.mcsi.ro/Plati_MCSI.

