**Partes del trabajo**

Front

Summary (one page)

Index

Introduction

Objectives (incluyendo descripción del problema, estudio de alternativas y

metodología empleada).

Technological description (puede incluir especificación, diseño e

implementación).

Conclusions (incluyendo los logros principales alcanzados y posibles

trabajos futuros).

Bibliography

Appendix

**Summary**

This document contains the description and development process of the Moodle Web Services Python Library. Firstly, it will be introduced the project itself with its motivation, its field, characteristics and its main objectives. Then, the document structure will be seen, listing each one of the chapters and summarizing their contents.

This is a summary of project’s structure:

* In first place, the starting Moodle's development point will be seen, which are his characteristics and what can be done with them.
* After that, it will be analyzed what kind of programs usually use Moodle as an external application and how they interact with it to work.
* Later, this python library solution will be explained. Also it will be said how external applications interacting with Moodle can be developed.
* Finally this library development process will be seen, his API and some examples of its use.

**Introduction**

This chapter summarizes preliminary information that the reader must acquire for a complete understanding of this document.

Throughout this document Moodle’s requirements, its main characteristics will be seen, its web services plug-in, the REST protocol sentences to interact with it, the python code to do it and diverse functionality provided by this python library.

**Project motivation**

Moodle has become a popular teaching support system all around the world. This and the increase of mobile capacities to interact with many systems in the web have made appear new applications for Moodle.

Because of this, Moodle developers decided to include some additional functionality to allow external applications work with Moodle since version 2.0.  
These functions are implemented as a Moodle plug-in, and its functionality initially was limited.

Later, the available functions list was extended on version 2.5, so you can do many things that Moodle can do but with this external service (for example, create courses, create group users or get submitted files).

However, the use of this Moodle functionality requires some advanced knowledge about Moodle functionality and REST or other applications data exchange protocols.

This project tries to give some tools to configure Moodle easily to get this functionality with a python library. This will provide some abstraction to users, so they can ignore Moodle’s communication process.

**Project Field**

The project is a Moodle Web Services Python Library, so the project fields are some Moodle parts (users’ roles, web services plug-in, enable REST protocol, API documentation, etc.), python code to connect with Moodle and how to make requests to get Moodle’s correct answers. Also it will be seen how to get answers in JSON format, to facilitate the data extraction.

**Project Objectives**

This library has been made to provide some external functions to facilitate the development of applications that interact with Moodle and to exploit this Moodle functionality.

However, Moodle system must be configured properly in order to that the library works correctly. So, a part of this memory is reserved to explain how to configure Moodle to work with this library.  
Also, to use this library you need to have accounts in your Moodle site, and users’ permissions changes this library behavior. In this way, it will be seen these permission repercussions over the library functionality.

Even though you have to take into account some factors in the Moodle’s configuration to use this library, this is not a Moodle use tutorial, so, only Moodle’s necessary sections to understand this project will be told.

Furthermore, the data structures returned by this library functions will be analyzed so the users will know how to obtain and manipulate the information they want.

**Memory sections**

Like it has been said, one goal of this project is to use the Moodle system with external applications.

To do this, the first step is study how to use Moodle. Logically, to do that first we have to install this Moodle system. But this is not an objective this project, so this part will be summarized and referenced to some external official documentation, where this installation process is explained in detail.

Then, Moodle must be configured so it can receive and answer requests from other applications. In this step it will be seen the different protocols that can be used to do this, but after that we will assume that the protocol used will be the REST protocol. On that time, it will be analyzed Moodle’s permissions, administrated with system roles for users. It will be sown how these permissions affects at answers given by Moodle too.

Once Moodle has been adequately configured, study of communication process will be started.  
First it will be shown the options to get information from Moodle, and some application examples which use these options.  
One of these options is the web services plug-in. This will be the service used to interact with this python library.

There are 2 parts in this communication process. The first one is the authentication process, where a user has a password associated to a web service with some specific functions. The second one is the request itself, and the response will depend of the user capabilities.

After that, the request and responses involved in this communication will be studied, what data needs the web service and what data returns. In this step it will be show how to do that responses will be in JSON format, in order to facilitate the data extraction in python code.

Once this is done, web services API will be analyzed, and will see how to create some functions in python to do the library. These functions will have the same functionality than the functions provided by the Moodle’s web service.

Lastly we will see the library’s API, the functions implemented, their functionality, the parameters necessaries to use them and the data returned by these functions.

**What is Moodle**

Moodle is an open source code system used for teaching and learning purposes. It allows creating on line dynamic web pages, creating courses and managing it with the web interface. This made it popular between teachers around the world as a tool to provide resources to students and support teaching. It needs a data base system, a web server and PHP to work properly.

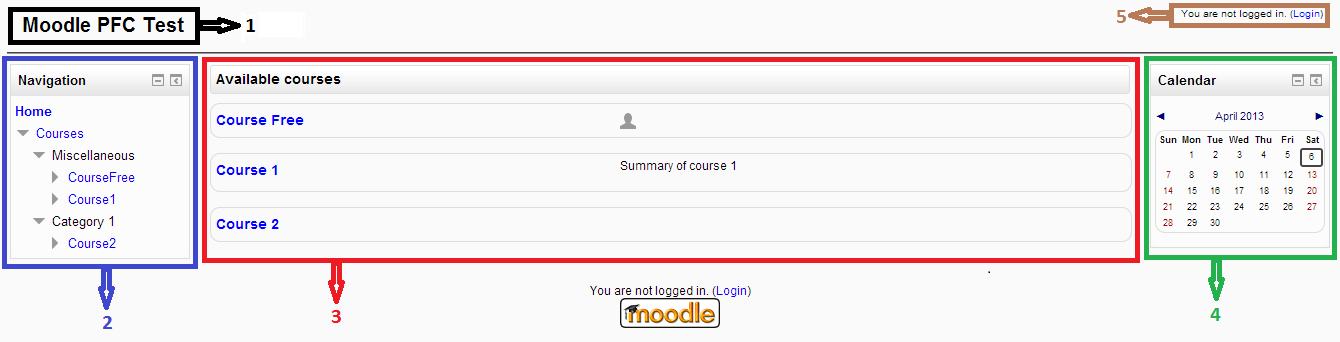
Moodle can work with several data base systems and web servers. Supported data base systems are:

* MySQL (minimum version 5.1.33)
* PostgreSQL (minimum version 8.3)
* MSSQL (minimum version 9.0)
* Oracle (minimum version 10.2)
* SQLite (minimum version 2.0)

Recommended web servers are Apache or IIS. The minimum version of PHP is currently 5.3.2.

<http://docs.moodle.org/23/en/Installing_Moodle>

**Moodle’s web interface**

Once we have installed Moodle we see its main page in our web browser. It will looks like this:

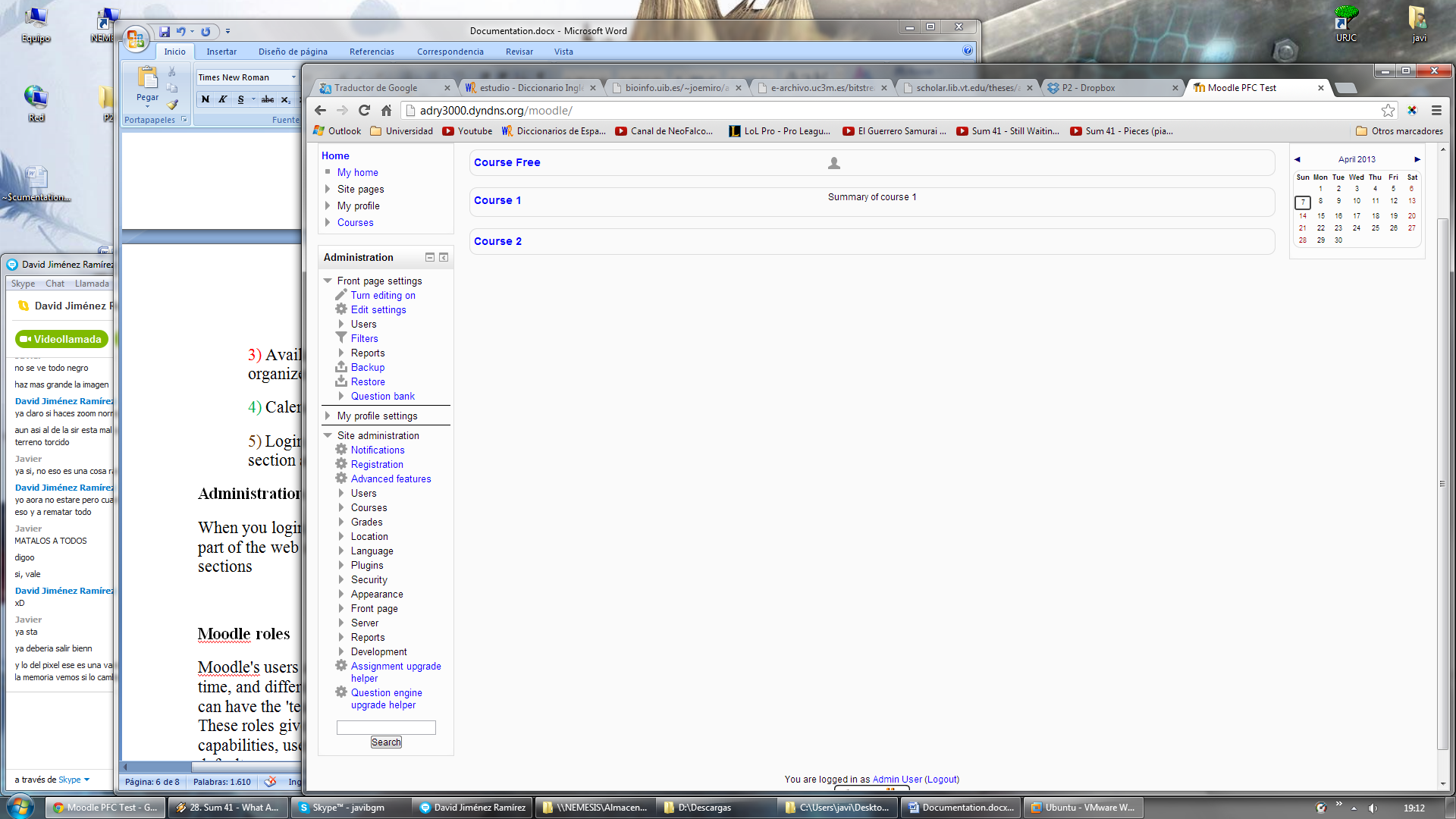
The fields marked are:

1) Moodle’s web site name

2) Navigation panel. With this you can navigate through categories and courses. An administrator can add more resources and panels to this main front page.

3) Available courses on this site. This section can be configured to show courses organized by categories.

4) Calendar panel.

5) Login user information. Once you have logged in you can see your name on this section and access to your user information.

**Administration panel**

When you login with a Moodle administrator account you can configure and modify every part of the web site. You can do this whit the administration panel. It is divided in 3 sections:

* With the first section you can configure the web page in which you are.
* The second one is your user profile settings. This section is available for every registered user.
* The last one is where you can configure every part of the Moodle web site and this is what interests us.

**Moodle roles**

Moodle's users are classified by roles. One user is able to have different roles at the same time, and different roles depending on the Moodle's part that he is. For example, one user can have the 'teacher' role in one course and the 'non editing teacher' role in another course. These roles give to users different capabilities on the system. Depending on these capabilities, users will be able to do some changes and consults to the system or not. By default, a user without role doesn’t have any capability on the system, so he can't do anything. The roles are what give the users the chance to do things in Moodle. There are some roles predefined in Moodle:

* Manger
* Course creator
* Teacher
* Non-editing teacher
* Student
* Guest
* Authenticated user
* Authenticated user on front page

These roles can be assigned to work on these contexts:

* System
* User
* Category
* Course
* Activity module
* Block

A category contains courses, and a course contains different type of activities separated in blocks.

<https://moodle.org/about/>

Moodle requirements:

* PHP
* Database system: MySQL
* Web server: apache

Moodle Web Services:

Moodle web services are additional Moodle modules that let you to use Moodle with other protocols than http such as AMF, SOAP, REST and XML-RPC. These services have been tough for interacting Moodle with other applications.

**Creating an external application using Moodle via Web Services**

* Moodle requirements:  
  REST protocol will be used in this application.

Using web services: <http://docs.moodle.org/24/en/Using_web_services>

To allow users to create and see their own security keys (token) you must assign a system role with these user capabilities:

* + users [moodle/webservice:createtoken](http://docs.moodle.org/24/en/Capabilities/moodle/webservice:createtoken) (if not, the administrator must create the token manually for each user for the service)
  + [webservice/rest:use](http://docs.moodle.org/24/en/Capabilities/webservice/rest:use)
  + service required capability (depends of the service, check Settings-> Site administration-> Plugins-> Web services-> Manage services->Edit)
  + Required capabilities for the service functions.
  + Also you must enable the web service function documentation, in Settings-> Site administration-> Plugins-> Web services-> Manage protocols
* Get web services token:  
  <http://docs.moodle.org/dev/Creating_a_web_service_client>

Service short name info: <https://moodle.org/mod/forum/discuss.php?d=197187>

* Get JSON answers to REST requests: <https://moodle.org/mod/forum/discuss.php?d=204469>

In the request you must include the parameter "moodlewsrestformat=json"  
example: /webservice/rest/server.php?wstoken=1cfc5fsd5a6fa75dfa&wsfunction=core\_webservice\_get\_site\_info&moodlewsrestformat=json

**Errors (check spelling)**

You could get different types of errors, grouped in 2 categories:

* Library exceptions: it will get this kind of errors as a TypeError python exception when it is used wrongly one of the functions provided. For example, if a function needs a list and you give an integer as parameter to this function, you will get a TypeError exception. In the same way, if the request needs specific parameters and these are not provided, a KeyError exception will be thrown. However, if a function needs, for example, some parameters inside an array to work properly and these parameters are wrong it will be received a next type error, because this is not checked by the library.
* Moodle exceptions: once the library has processed the data provided to use a function, the request to Moodle site is done. However, you still can get an error by many reasons: the data requested doesn’t exist on this Moodle site, the requested parameters have wrong data types, are missing required parameters, etc. On this situation, Moodle will get an error message, and the library will throw a ValueError python exception specifying found problems.

**Bugs (check spelling)**

Function: get\_calendar\_events.

Bug: when you get global site events, only get global events if they had been created by the python library. Those created manually via web interface are not recognized.

Function call: MoodLib.get\_calendar\_events(userevents=0)