­­­­­­

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA

DE TELECOMUNICACIÓN

DOBLE INGENIERÍA DE TELECOMUNICACIÓN E

INGENIERÍA TÉCNICA EN INFORMÁTICA DE SISTEMAS

**PROYECTO FIN DE CARRERA**

TÍTULO

Autor: Javier Benito García-Mochales

Tutor: Jesús M. González-Barahona

Curso académico 2012/2013

**Partes del trabajo**

Front

Summary (one page)

Index

Introduction (resumen de lo que se va ha hacer en el proyecto y la estructura seguida en la memoria)

Objectives (lista de 5 o más objetivos genéricos, sin demasiado detalle).

State of the art: descripciones brebes de todo lo impicado en el proyecto

Ámbito del proyecto   
 Todo de lo que se haga(que es un servidor web, apache)  
 También cómo interactúa, con REST y otras tecnologías, XML-RPC, JSON, XML <http://es.wikipedia.org/wiki/Remote_Procedure_Call>  
<http://es.wikipedia.org/wiki/XML-RPC>  
<http://es.wikipedia.org/wiki/SOAP>  
Qué es el Moodle y sus semejantes (LMS) <http://en.wikipedia.org/wiki/Learning_management_system>  
 Contar los elementos que tiene Moodle, sus usos y elementos básicos (que es un usuario o un curso o un fichero en Moodle y como se relacionan)  
 Tambíen como el Moodle está separado en módulos y su descripción  
 Hablar del scrapping en general, no solo para Moodle <http://en.wikipedia.org/wiki/Web_scraping>  
 Características generales de python, sus bibliotecas (las usadas) y otros lenguajes.  
 Contar las tecnologias, tanto las que he usado como las que no, de forma independiente al proyecto(eclipse, entornos de desarroyo, PHP)

Technological description

Requisitos: descripción del problema

Diseño: decisiones de diseño, porque se ha usado una cosa y no la otra (deben estar explicadas por encima en estado del arte, los detalles del porqué se toma esta decisión en este apartado). Estudio de alternativas

Metodología de desarrollo: desarrollo en espiral

Implementation: siguiendo las fases en espiral (por ejemplo, las siguientes)

1 – Ver como funciona Moodle y la api de Moodle  
 2 – Como hacer para usar los web services  
 3 – Pruebas y ejemplos de uso  
 4 – Problemas encontrados  
 …

Results and conclusions (incluyendo los logros principales alcanzados y posibles trabajos futuros).  
 Resultados de verdad: biblioteca con funciones dadas, cada funcion sería un resultado. Entre los resultados deberían estar los objetivos del proyecto.  
 Todo lo que he hecho: instalar un Moodle, aprender a manejarlo, aprender a usar la API de Moodle (falta de documentacion), uso de herramientas para documentación.  
 Trabajo futuro: cosas que se podrían hacer para continuar con este proyecto y otras cosas que se apoyen en el uso de esto.

Bibliography

Appendix (o anexos)  
 Donde está todo el proyecto (github), para poder obtenerlo y como instalarlo/ configurar para usarlo  
 API de mi librería  
 Ejemplos de uso

**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 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.

Index

1. Introduction 6

1.1. Project motivation 6

1.2. Project Objectives 6

1.3. Memory sections 7

2. State of the art 9

2.1. Project Field 9

2.2. What is Moodle 15

3. Technological description 21

3.1. Requirements 21

3.2. Design 21

3.3. Development methodology 26

4. Implementation 27

4.1. Install Moodle and work with its web interface 27

4.2. Activate and configure web services: enable web services, create a web service, add service short name to database. 29

4.3. Get token, login with web service 30

4.4. Use some functions: test with courses functions 30

4.5. Plan library structure, study API structure, start library documentation 30

4.6. Develop short programs using this API 30

5. Results and conclusions 32

6. Bibliography 34

7. Appendices 34

7.1. Appendix 1 34

1. Introduction

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

Throughout this document the Moodle requirements, its main characteristics, 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 will be seen.

* 1. Project motivation

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

Because of this, Moodle developers decided to include some additional functionality to allow external applications can work with Moodle since version 2.0.  
These functions are implemented as a Moodle plug-in, and its initial functionality was limited.  
Later, the available functions list was extended on version 2.5, so many things that Moodle can do with its web interface, can be done through this external service (for example, create courses, create group users or get submitted files).

However, using 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 the Moodle communication process.

* 1. 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 do 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 it is necessary have accounts in the 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 configuration to use this library, this is not a Moodle use tutorial, so, only necessary Moodle 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.

The objectives of this project are summarized on these points:

* Analyze Moodle parts necessary to configure the system so this library can be used adequately.
* Create Moodle web services with every function provided by the Moodle site.
* Create the library with the functions necessaries to use the Moodle web services.
* Define necessary data structures to use these functions and the structures returned by these functions in order to work with the Moodle data.
  1. Memory sections

As stated above, one goal of this project is to use the Moodle system with external applications.

To do this, the first step is to study how to use Moodle. Logically, to do it this Moodle system has to be installed firstly. But this is not one of the project objectives, so this part will be summarized and it will be referenced to external official documentation, where this installation process is explained with more detail.

Then, Moodle must be configured so it can receive and answer requests from other applications. On this step different protocols that can be used to do this will be seen, but after that, it will be assumed that REST protocol will be used. On that time, Moodle permissions will be analyzed, which are determined by system roles assigned to users. How these permissions affects to Moodle answers will be shown too.

Once Moodle has been well configured, the study of communication process will be started.  
First it will be shown the options to get information from Moodle, and some application examples that 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 it will be seen how to create some functions in python to make the library. These functions will have the same functionality than the functions provided by the Moodle 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.

1. State of the art
   1. Project Field

The reader must have some basic knowledge about web systems, programming concepts and other informatics notions to understand this project domain. In this section these general basis will be explained without much details, focusing on those that will serve to explain the development of this project later.

* + 1. Web servers

Firstly, a web server is a system that runs programs to generate a response to client’s requests through the Internet. The main purpose of a web server is to deliver web pages to provide any kind of information.

Usually, the users interact with the web servers with a web browser, which one communicates with the web server with the HTTP protocol. The web server answer to the web browser with the user’s request response with a HTML file and it is interpreted by the browser to do that a human can show the information provided.  
However, servers can have other functionalities besides the web pages services. There are database systems, storage systems, distributed revision control systems and many others. Also, there are many applications besides web browsers that do requests on the Internet, and they use other protocols instead of HTTP to interact with the server. The reason of that web servers are the most known is that every server usually has a web interface where users can register and get details of how to use the system they are using.  
Some of the most important web servers are Apache, Internet Information Services (IIS), Cherokee and Tomcat.

* + 1. Database systems

The database systems are important in this context too, because their mission is storage of big quantities of data information. These systems are composed by two parts: the database itself, whose purpose is the efficient storage of the information and the database management system in charge of administration and to provide the database information to users and other applications. The database management system defines the database structure, which can storage data in many different ways.  
Some examples of database systems are MySQL, PostgreSQL, SQLite, Microsoft SQL Server, Microsoft Access and Oracle. Each one of those systems has different definitions of database and one created with a database system cannot be used by other database system. Nevertheless these systems can operate between them and with other applications using some standards like SQL and ODBC or JDBC.

* + 1. Communication protocols

There are many protocols that can be used to communicate two processes over the Internet. We have already mentioned the HTTP protocol, used by the web browsers and it can be used by other protocols to exchange information. This protocol is based on the REST (Representational State Transfer) software architecture technique and systems that follow this architecture are known as RESTful systems.

Some REST systems main characteristics are:

* Clients and servers are separated in development terms so clients do not need to know what servers do with their requests and vice versa.
* They do not have a protocol status because each request and response contains all the information necessary to understand it.
* They have only four different well defined operations: POST, GET, PUT and DELETE.
* They identify each resource unequivocally by his URI (Uniform Resource Identifier).

Moreover, many applications work executing some code in a remote machine and it returns the result of that execution. For this, the RPC (Remote Procedure Call) protocols were developed, which provide a wide variety of functions to do that. An example of these protocols types are the XML-RPC or its successor SOAP. On these protocols are defined an extensive IDL (Interface Description Language) with all the operations that servers with that protocol can do in contrast to the RESTful systems, which only have four possible operations.

* + 1. Data structures

Some information exchange protocols have been said but the information has to be formatted too, depending of what formats can servers and clients support.  
Those formats define the structure of the contained data, so the receiver knows the data and its meaning. Some example of this data structures are XML, JSON or HTML.  
XML and HTML are markup languages that define how documents must be structured to assign at each kind of data their corresponding mark. These marks are named labels or tags. These both languages generate documents that are simultaneously human-readable and machine-readable.  
HTML is used in the web navigation through Internet, web browsers do requests to an URL and they receive a HTML document that they interpret with those labels to show the data in a nice way for a human.  
XML is used as language to define data structures to exchange information in many contexts because of its simplicity and because it is an extensible language. This means that with this language you can combine data of different languages, allowing a general use in many platforms.  
Another difference between them is that a non well-formed HTML document can be interpreted by a web browser, but the XML is focused in creating only well-formed documents.

JSON (JavaScript Object Notation) is known as a light data exchange format which comes from the JavaScript programming language. It transports programs data with the same structure that is used in the JavaScript programs. This provides a simplicity that improves the data transfer process and facilitates data extraction process because it do not use tags like XML or HTML. This and the expansion of JavaScript to all modern web browsers have made that JSON is being used in many contexts, especially as substitute of XML.

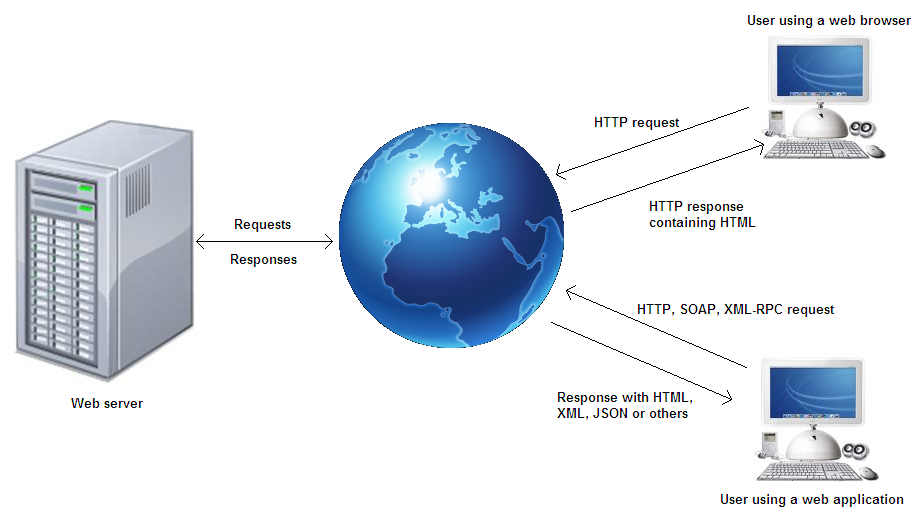


Figure 1: Web communication schema

* + 1. Parsing

Parse a sentence consists in separate its components and analyze them syntactically. In computing, parsing it is applied to text strings obtained from different sources, for example from a file. This analysis separates the information contained and metadata, which represents the data structure. Once its data and its structure are separated it can be analyzed to detect syntactical problems. Interpreters and compilers do this parsing process to check that the code has a correct syntax according to its programming language.

* + 1. Scrapping

The "scrapping" is the technique of extracting original data from processed data, using reverse engineering for retrieving original data used to generate the document which is being analyzed for this data extraction.

The web scraping does this data extraction from websites and it is used by many programs. These programs interact with a website like a normal user which uses a web browser, and analyze the HTMLs returned by the web sites to extract the necessary data which they need to work. work with them more efficiently, storing the data in a data type known by the programming language used.   
These programs are known as "bots", and they can do requests to the web much faster than a normal user. Some web servers try to restrict their access because it can be harmful for the server (for example a Deny Of Service (DOS) attack)

* + 1. Programming Languages

All these applications that exchange data work to meet its usefulness depending on their programming defined in their source code. This source code can be wrote in many programming languages, each one of them with different characteristics. Those languages are used to give instructions to a machine, usually to a computer, and determine their behavior to resolve problems, obtain information, or anything else that can be done with a computer. Those languages can be interpreted or compiled, can have different programming paradigms (object-oriented, imperative, functional, declarative, etc), have their own libraries and many other properties that differentiate them. Those properties make that programmers have to think on one way or another to do programs depending of the programming language used. There are many programming languages and many variations of each one, but actually some of the most important and that will be used in this project are PHP, JavaScript and Python.

PHP is a free scripting language designed for web developing on the server side but it can be used to develop any other type of applications. It is an interpreted, imperative, functional and reflexive programming language and it started being object-oriented since its third version.   
It usually works on the server side on the web navigation because of its capacity of generate HTML files with embedded code without needing to process the code with external functions. This means that a HTML can have PHP code inside it and the PHP processor executes the code to include its results in the HTML returned to the user. However, since its beginning, it has been extended to include other programming paradigms and it has become a general-purpose programming language. [[1]](#footnote-2)

JavaScript, like PHP, is a programming language used for creating dynamic web pages but in this case, on the client side. Despite its name, JavaScript has no relation with the famous programming language Java. It was developed by other people and for different purposes.  
Its interpreted code allows to web servers to return web pages with JavaScript code embedded, which can be interpreted by any modern web browser. JavaScript code allows creating dynamic effects such as showing or hiding buttons or text, creating animations, showing warning messages, everything without the server participation.[[2]](#footnote-3)

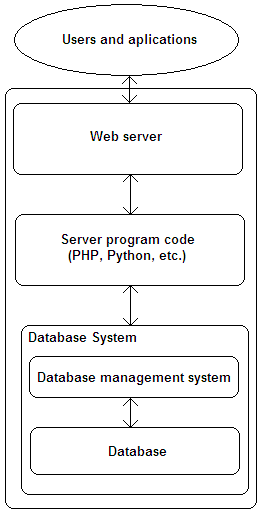
Python is an object-oriented, interpreted, and interactive programming language. It is a high-level language capable to design programs with much fewer code lines than many other programming languages. Another of its properties is that it generates easily readable source codes because of the language specification. Also it has a complete and understandable standard library. All this characteristics and its multi-paradigm programming have made it a general-purpose programming language, used in both scripting and non-scripting contexts.  
Because of this, Python language was selected to develop this project library.

Figure 2: Server structure

* + 1. Integrated Development Environment

An IDE (Integrated Development Environment) is a program that provides several tools to help programmers to develop applications.

Some of those tools are a source code editor, that usually suggests the possibilities to autocomplete the code that is being written or shows some possible fixes to a code problem found, automatic build tools and a debugger, that lets find execution time problems showing system status after every code line executed.

Some IDEs are prepared to develop programs from a specific programming language, but they can be multi-language by default or adding external packages. Some popular multi-language IDEs are Eclipse, Oracle JDeveloper, NetBeans and Microsoft Visual Studio.

* + 1. Learning Management Systems

A LMS (Learning Management System) is a software application used for teaching and educational purposes. Those systems provide a platform where students and teachers can exchange information about their courses, resolve problems with forums, share files to complete the teachings or to evaluate works made by the students, do online evaluations and much more. These systems usually have a web interface to facilitate access and administration tasks, they allow registered users to enter and these users are registered in different courses, where they can access. The courses can be created and modified by administrators, managers and teachers, where they put contents for the students. Another characteristic of these systems is the capacity of tracking the students’ progress and report their activities or analyze their skills.  
The LMS term is frequently misused to refer a similar type of system such as CMS (Course Management Systems) or LCMS (Learning Content Management System). These ones are more related with development and creation of contents, unlike the LMS that focus in administrate the contents in courses structure and provide other tools for teaching support taking external data contents provided by the users.  
LMS systems are both pay and free, the most used LMS in 2011 is the non-free Blackboard Learning System, followed by the Moodle system, a free software platform.[[3]](#footnote-4)  
Those both characteristics, be free software and be popular, made the LMS Moodle was chosen to develop this project.

* 1. What is Moodle

As noted above, 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.

* + 1. Content Management

Moodle organizes his resources in a hierarchy structure. Its main courses structures are the categories, which can contain several courses and other sub-categories. These categories can represent a whole university career, the courses of one year of studies or a set of courses to reach some goal.  
A course always has to be in a category or a sub-category, and it contains all the course resources divided in sections. These sections are used to set a course format. This format can be weekly, split in topics, to show social forums and for using SCORM (Sharable Content Object Reference Model), a set of standards and specifications to share contents between web-based educational systems such as the LMSs.  
The resources can be items such as books, Files, Folders, URLs, etc. or activities such as assignments, chats, forums, tests and many other things.

* + 1. Authentication and roles

Usually, users need to login in Moodle to access its contents, and each user can be restricted to access only to some contents. A non logged user can use the Moodle guest account to see public contents, which are in courses that allows the guest account access.  
The users’ restrictions are established by their user roles. A user is able to have different roles at the same time and their roles can be different depending on the Moodle part where he is. For example, a user can have the 'teacher' role in a course and the 'non editing teacher' role in other course. These roles give to users different capabilities on the system. Users will be able to do some changes and consults to the system or not depending of these capabilities. By default, a user without role does not have any capability on the system, so he cannot do anything.  
There are some roles predefined in Moodle:

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

More roles with the specified capabilities can be created and assigned by the site administrators. The roles are assigned in different context. The context is what determines the roles that a user have in a page of the Moodle site or another.  
These role contexts are:

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

Once a user is logged in, he can access to those courses where he is enrolled. Normally, users can self-enroll themselves or can be enrolled by an administrator but there are many other enrollment methods. Users must be enrolled in those courses that they participate, so they receive information about those courses and not from every course created in the Moodle site.

* + 1. Moodle modules

The Moodle core is programmed in PHP, and it is split in different modules and blocks, which represent Moodle parts. Each module is responsible of its own functionality so all Moodle modules can interoperate between them. For example, every action done in a forum, such as creating a new forum or write a message on it, must be programmed in the "forum" module. External modules can be developed and implemented to expand Moodle functionalities. Some Moodle modules are:

* Blog: allow users to create personal blogs
* Calendar: shows a calendar panel in the browser and can create different kind of events in its dates.
* Course: manage the control of categories and courses structure
* Group: allows creating users groups and creating groupings of these groups.
* Message: provide a messenger service, with a searcher of users and a contacts list to exchange messages between users.
* Notes: creates personal notes about users
* User: manage Moodle users and their information.
* Web service: provides some Moodle functionalities for external applications.

This project will be focused on this last mentioned module, which is able to interact without using the web interface.

* + 1. The Moodle web interface

Figure 1 shows an example of a Moodle main web page shown in a web browser.

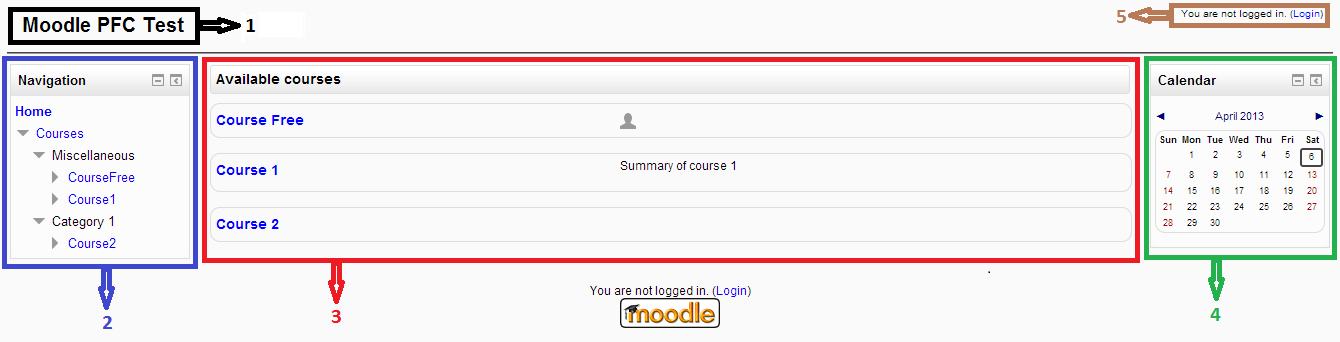
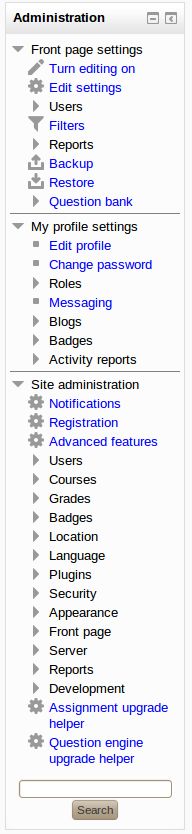


Figure 3: Moodle front web page

Marked fields are:

**1)** The Moodle 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, one of the Moodle modules.

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

This main page can be modified by an administrator user to show other contents or panels. In the same way, administrators can change many other Moodle options with the administration panel.

* + 1. 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 with the administration panel. It is divided in 3 sections:

* With the first section it can be configured the actual web page. In this case, the front page can be edited but it could be a course, a user or other Moodle element.
* The second one is the user profile settings. This section is available for every registered user.
* The last one is where it can be configured every part of the Moodle website and where it can be done most of the changes.
  + 1. Moodle Web Services:

Figure 4: Administration panel

The Moodle web services are additional Moodle modules included in its version 2.0. They let you use Moodle with other protocols than HTTP and they have been thought for interacting with other applications. They can be activated and configured in the "Plugins" section in the site administration panel.

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 the Moodle correct answers. Also it will be seen how to get answers in JSON format, to facilitate the data extraction.

1. Technological description

Requisitos: descripción del problema

Desarrollar aplicaciones que obtengan informacion y actuen sobre moodle: parsing, Mobile web app y web services

Diseño: decisiones de diseño, porque se ha usado una cosa y no la otra (deben estar explicadas por encima en estado del arte, los detalles del porqué se toma esta decisión en este apartado). Estudio de alternativas

Metodología de desarrollo: desarrollo en espiral

* 1. Requirements

Provide a list of functions to interact with Moodle. Those functions must let do all functionalities given by the Moodle web services in the Moodle version 2.5.

* 1. Design

In this section it will be seen the design decisions taken for this library developing and why those decisions have been taken.

Python

* The Python programming language will be used because is facile to learn, necessarily generates easily readable code and its diverse documentation, examples and libraries for being open source.

Eclipse

* Eclipse is the IDE chosen because it is a free software development tool, commonly used and easily usable. Eclipse is a Java applications development tool by default but due to its fame many external plugins exist so it can be used for applications development on other programming languages. As it has decided that python will be the programming language used in this project, one of these external Eclipse plugins will be necessary. The Eclipse plugin used is called PyDev and provides all IDE basic functionalities for python such as code completion with auto import, syntax highlighting, code analysis, go to definition function, refactoring, mark occurrences, debugger and many others.

JSON

* JSON format will be used for receiving answers from the Moodle website. Its syntax is very similar to the basic Python data structures, and the standard library provides a module to pass data in JSON format to these structures.

REST

* To interact with Moodle system it will be used the request system defined by REST architecture, using the POST method to give to Moodle the request with necessary parameters. As it was explained above, this kind of request/answer protocols are commonly used on the Internet, like the HTTP protocol. Also, these protocols do not have status, so each request is independent from others, so clients and servers do not need to store session records. In addition, the Moodle web services system supports this type of data transmission protocol.

Web services

On this sub-section it will be seen the different kind of applications for Moodle and how they work to get the information they need to work. After that it will be shown the solution to this problem chosen in this project and other possible solutions.

Normally, users get the information they want from Moodle by its web interface with a web browser. Web applications can be classified in two big groups depending on how they obtain these data. They can use the scrapping technique, which was explained on the section "state of the art", doing requests as a normal user and extract the information analyzing the HTML file returned by web server or they can exchange information with other available systems in the server through other protocols.

1. **Scrapping HTML applications**

Moodle web services are an additional functionality added to Moodle on its version 2.0 and maintained on later versions. However, Moodle applications development started before this plugin exist, so they cannot use this system to interact with the Moodle system. Instead of this, they usually parse the HTML gotten from a normal HTTP request and use the scraping technique to get the information they want.  
Those applications usually search through the HTML file some specific strings or HTML *tags* to get specific information, like the tag used to get the user’s name logged in or the courses shown in the main page. They take the links necessary to navigate in the Moodle site from this HTML, so they get the information like a normal web explorer would do.  
With this, the applications can work without any additional system and be compatible in different web pages without configuring anything.  
However, those applications have some important problems. One of them is that they could not work in some Moodle places because of changing their appearance or version will affect to the HTML received. This could cause that those strings used to search the information necessary to the application will not be found, and make a malfunction of the application.  
Also, the HTML processing can be very complex in some cases, with all the problems that this processing could bring.  
Another disadvantage is that each request gets the entire HTML file so much information is dropped because an external application do not need it (for example, the information used to show the elements in the web browser or that related with the page visual design). This carries several problems like using more bandwidth than necessary and therefore more delay in the obtaining data process.

Besides, the Moodle site cannot differentiate between users navigating with normal web browsers and applications that work on this way. These applications can be detected with other systems which analyze net traffic and recognize these applications' behavior.

Look more: Application with HTML parsing: <https://github.com/praveendath92/MDroid>

1. **Using Moodle Mobile app**

Another option to do this is to use the Moodle Mobile app web service. This solution is similar to the chosen one in this project. As its name indicates, this option uses a special Moodle web service developed for mobile applications. This service has some predefined web services functions and is available in the Moodle site by default since the Moodle web services were implemented. However, like every web service, the web services option must be enabled in the Moodle site to use it.

This web service functions are unchangeable and limited so not every web service function can be used with this service. Because of this, in this project is decided to use generic Moodle web services, with which ones you can add every function available and exploit all this Moodle plugin.

<https://play.google.com/store/apps/details?id=com.moodle.moodlemobile>

1. **Using other web services**

Using Moodle web services instead of web scraping has its advantages and its disadvantages. If web scraping is used a deep analysis of the HTML files is necessary, but if web services are used then it is required to learn how to use those services. To do that, the web services API must be studied, just like the way how to make requests with the protocol enabled to communicate with the server. In this API there are all the web services functions names, the parameters they need to execute, these parameters types and the data structures returned by these functions. The wrong use of these functions will make that the Moodle service returns an error.  
Once this API is studied a determined web service must be created and configured with desired functions to use them. The available functions that this service could provide depend of the Moodle version installed and other external plugins which can add non-official functions.  
These functions can be used in an external program and will work in the same way in future versions. However, if web scraping is used the potential applications functions will not depend of the web services plugin, it depends of what can be done with the scrapping process.

A library similar to the one done on this project: <https://github.com/zikzakmedia/python-moodle>

* + 1. Library structure

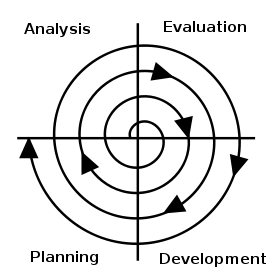
Looking the Moodle web services API it can be seen that its functions are separated in modules depending on the Moodle parts affected. Functions which change or take courses information are defined in the Moodle core module for courses. This can be detected looking the functions names. So in order to develop this library it has been decide to follow the same schema, keeping this modules structure. Each module has a Python class with the functions that affect to the corresponding Moodle components, so, for example, functions which affect to a course are in the course module.

All functions have some common code, as the code to connect with the Moodle site. What changes at using one function or other are the provided parameters, so this common code is defined in a superior module which can be used by all modules and it is called MoodClass.

Also it is defined a module which can access to all modules so it can use all defined functions. With this module called MoodLib, it is not necessary to call a specific module to use a function of that module because they all are accessible from the MoodLib module.

So the next figure represents the resulting structure following this modules schema:

* 1. Development methodology

[[4]](#footnote-5)This project has followed a spiral development methodology, which consist in a set of tasks that are repeated in an order in different cycles to incrementally develop the library. Each cycle consists in four steps:

1. Determine cycle objectives: in this phase it have to be determined several tasks that would let reach the final goal in future cycles. It is important to seek all possible ways to complete these tasks in order that the best choice can be chosen. Also it is necessary to have well defined the cycle scope and its limitations depending on the dedicated time to that cycle.
2. Risk analysis: predictable problems are analyzed to evaluate their cost and decide if resolve those problems and how resolve them, or search other alternatives to avoid those problems.

Figure : A software development spiral.

1. Developing and testing: this phase consist in making those tasks programmed in the first phase with the solutions provided in the second one. Before finishing this phase it must be ensured that all products works correctly and as they are defined in the initial phases.
2. Planning of next phases: once this cycle objectives are reached, new targets must be defined for future cycles taking into account the time expended and the knowledge acquired in this cycle.
3. Implementation

This section explains the whole developing process followed in the project, indicates the steps followed to create and test the library and the way to use it. Each one of these steps represents a developing cycle of the followed spiral methodology.

* 1. Install Moodle and work with its web interface

The first step to develop Moodle applications is to have a functional Moodle to use and test these applications. So firstly it is necessary the Moodle installation. Moodle is a system that needs other sub-systems to work, and they must be prepared and configured before installation Moodle process is started.

The Moodle core is programmed in PHP and it needs an interpreter because it is an interpreted language. This interpreter must be installed where the Moodle system will be working so the computer can understand and execute the PHP code. This library works with the Moodle 2.5 version and the minimum PHP version for this Moodle version is the 5.3.3.

It also needs a database system to store all information relative to the Moodle site. This information can be users data, courses contents, stored files, students grades and many others kinds of data records. Moodle core system is programmed to work with several data base systems. These are the supported data base systems:

* MySQL, minimum required version: 5.1.33
* PostgreSQL, minimum required version: 8.3
* MSSQL, minimum required version: 2005
* Oracle, minimum required version: 10.2

A last additional system that Moodle needs its a web server. This is the system part that will take over the communication between Moodle and its users, with the web browser or with other web services. The web server is the interface that connects the Moodle site with the Internet, making it accessible from everywhere. Fully supported web servers are Apache and IIS, and they must be configured so they can serve PHP files. “The version is not critical but try to use the newest web server build available to you.”[[5]](#footnote-6)

Once those systems are installed it must follow some steps to prepare the Moodle installation.  
The Moodle files must be downloaded and put in the web server location so it can be accessible with a web browser.  
After that, a new empty database must be created in the database system and configured in order to that Moodle can to access and store its information on it. In this process it must be remembered four parameters necessary so the Moodle site can to manage the database. These parameters are the database server hostname where is the database system, the database name, the username to access at the database and the password for the above user.  
Then, it must be created a directory where uploaded Moodle files will be stored. The directory permissions must be changed so Moodle can access and write on it.  
Finally, the Moodle installation process should be started using the command line installer or the web based installer. Both options guide through the installation and request information for basic Moodle configuration.

<http://docs.moodle.org/25/en/Installing_Moodle>

With this, it has been created an empty Moodle site, with no courses, no resources, and only with the administrator user account created in the installation process. With the admin user can be created other users with permissions to manage the Moodle site, or create other admin accounts. With those accounts it can start to create categories to contain courses with the structure agreed. Those courses will be individually managed by their associated teachers, who will have their accounts configured for it. Students’ accounts can be created manually by managers or can be auto-created by the students if the self registration option is activated in the login page by an admin.

All images and references to Moodle parts will be done with the standard installation of Moodle version 2.5.

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

* 1. Activate and configure web services: enable web services, create a web service, add service short name to database.

Before using the Moodle web services there are several things that must be done and configured. These actions only can be performed by an administrator in the Moodle site and a user with permissions in the Moodle data base created for its use. Once the web service is activated, it can be used by every Moodle user.

Activating Moodle web services

The first step is activating the Moodle web services functionality in the site. For that, an admin must login and use the administration panel. In this panel, go to the "Advanced features" section and enable the option "Enable web services".

After that, it must be enabled, at least, one of the web services protocols according with the external application programming. In this case, the Python library is programmed for working with the REST protocol, so this is which must be enabled in the Site administration, Plugins, Web services, Manage protocols section. In this section, it is recommendable to enable the “Web services documentation” too, so users will be able to see what functions of the web services they are able to use depending of their capabilities. If a user does not have the required capabilities to use a function, it will not appear in his documentation page (this will be explained in more detail in the “Setting users capabilities” section).

Create a new web service

Once web services are enabled, a new web service will be created to use its functions. For this, go to Site administration, Plugins, Web services, External services section and click in the “Add” button. A new window lets to introduce the new service name and the “enabled” checkbox must be activated to use it. If more options are shown it can be seen two more:   
“Can download files” option must be checked if it is wanted that users could download files with web services. If this functionality is desired to be used with the library, this option must be selected.  
If the web services use want to be restricted to a group of users the "Authorised users only" option must be set and only the users with the capability selected in the "Required capability" list will be able to use this web service.  
Once the web service is created it can be seen in the External services section but it has no functions. So they are added with the functions link for the corresponding web service. Here there must be added all functions that want to be used. This library works with all non-deprecated standard functions for this Moodle version. To see the functions list supported by this library see Appendix 1. When all functions that will be used have been added to the web service each one of them can be seen in a list with a short description and the required capabilities that a user must have to use that function.

With the Moodle web service created on the website, it is necessary to access the database system and do some modifications. How to enter in the database depends of the database system installed before the Moodle installation but the necessary changes can be done with the same account that Moodle uses to access the system (and which was provided in Moodle installation process). Once inside the database system, the "external\_services" table must be found in the database used by Moodle. In this table it can be found information about the existing web services in the Moodle site and one of them should be the service created before. One of the parameters that can be found in this table is the “shortname”, and the new web service should have this field empty. If it is desired that users can obtain their token (string used for authentication in web services) using their username and password with the web services, a shortname must be provided to the web service.

User capabilities

* 1. Get token, login with a web service
  2. Use some functions: test with courses functions
  3. Plan library structure, study API structure, start library documentation

Web site string format: <http://mipaginaweb> y no mipaginaweb  
Get identifiers of users, courses, etc. from the URL  
Install pydev on eclipse  
Install epydoc for generating API documentation

* 1. Develop short programs using this API

**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) a system role with these user capabilities must be assigned to those users. These capabilities are:

* + 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, the web service function documentation should be enabled (in Settings 🡪 Site administration 🡪 Plugins 🡪 Web services 🡪 Manage protocols). With this, each user will have the documentation of the functions that they can use depending of their capabilities. If a user does not have the required capabilities to use a function, it will not appear in his documentation page.
* 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

**Moodle web services API**

All information about Moodle web services is available in Settings 🡪 Site administration 🡪 Plugins 🡪 Web services 🡪 API Documentation. This section can be seen only by administrators so they can see the functions they want to activate with the web services in the Moodle site. Normal users only can see the functions they are able to use according with their capabilities with de Documentation link in the Administration 🡪 My profile settings 🡪 Security keys page.  
Here can be seen every function, their description, the structure and description of their input parameters and their return values.

1. Results and conclusions

**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)

1. Bibliography
2. Appendices
   1. Appendix 1

List of web services functions that this Python library supports:

|  |
| --- |
| core\_calendar\_create\_calendar\_events |
| core\_calendar\_delete\_calendar\_events |
| core\_calendar\_get\_calendar\_events |
| core\_cohort\_add\_cohort\_members |
| core\_cohort\_create\_cohorts |
| core\_cohort\_delete\_cohort\_members |
| core\_cohort\_delete\_cohorts |
| core\_cohort\_get\_cohort\_members |
| core\_cohort\_get\_cohorts |
| core\_cohort\_update\_cohorts |
| core\_course\_create\_categories |
| core\_course\_create\_courses |
| core\_course\_delete\_categories |
| core\_course\_delete\_courses |
| core\_course\_delete\_modules |
| core\_course\_duplicate\_course |
| core\_course\_get\_categories |
| core\_course\_get\_contents |
| core\_course\_get\_courses |
| core\_course\_import\_course |
| core\_course\_update\_categories |
| core\_course\_update\_courses |
| core\_enrol\_get\_enrolled\_users |
| core\_enrol\_get\_enrolled\_users\_with\_capability |
| core\_enrol\_get\_users\_courses |
| core\_files\_get\_files |
| core\_files\_upload |
| core\_get\_component\_strings |
| core\_get\_string |
| core\_get\_strings |
| core\_grade\_get\_definitions |
| core\_group\_add\_group\_members |
| core\_group\_assign\_grouping |
| core\_group\_create\_groupings |
| core\_group\_create\_groups |
| core\_group\_delete\_group\_members |
| core\_group\_delete\_groupings |
| core\_group\_delete\_groups |
| core\_group\_get\_course\_groupings |
| core\_group\_get\_course\_groups |
| core\_group\_get\_group\_members |
| core\_group\_get\_groupings |
| core\_group\_get\_groups |
| core\_group\_unassign\_grouping |
| core\_group\_update\_groupings |
| core\_message\_block\_contacts |
| core\_message\_create\_contacts |
| core\_message\_delete\_contacts |
| core\_message\_get\_contacts |
| core\_message\_search\_contacts |
| core\_message\_send\_instant\_messages |
| core\_message\_unblock\_contacts |
| core\_notes\_create\_notes |
| core\_notes\_delete\_notes |
| core\_notes\_get\_notes |
| core\_notes\_update\_notes |
| core\_role\_assign\_roles |
| core\_role\_unassign\_roles |
| core\_user\_create\_users |
| core\_user\_delete\_users |
| core\_user\_get\_course\_user\_profiles |
| core\_user\_get\_users |
| core\_user\_get\_users\_by\_field |
| core\_user\_update\_users |
| core\_webservice\_get\_site\_info |
| enrol\_manual\_enrol\_users |
| mod\_assign\_get\_assignments |
| mod\_assign\_get\_grades |
| mod\_assign\_get\_submissions |
| mod\_forum\_get\_forum\_discussions |
| mod\_forum\_get\_forums\_by\_courses |

**DUDAS**

State of the art: hasta que nivel de detalle hay que explicar (hay que explicar lo que es un navegador web, por ejemplo)?

Requirements: nose que más poner

Design: Contar la estructura de la libreria aquí?

Alternativas a todas las decisiones de diseño? Nombrarlas o detallarlas?

Alternativa a Web services = scrapping

Alternativa a REST = SOAP, XML-RPC, AMF

Alternativa a JSON = XML

Implementation:

Instalacion de Moodle: poner requisitos de hardware? Poner requisitos de navegador web?

Referencias: como poner referencias a paginas web: como una cita? En anexos?

**POR HACER**

Explicar cada uno de los modulos y de los roles en la sección state of the art.

Cambiar (o no) los subtitulos de la seccion de Implementation

1. <http://www.php.net/manual/en/intro-whatis.php> [↑](#footnote-ref-2)
2. <http://en.wikipedia.org/wiki/JavaScript> [↑](#footnote-ref-3)
3. [*A Profile of the LMS Market (page 18)*](http://www.campuscomputing.net/sites/www.campuscomputing.net/files/Green-CampusComputing2011_4.pdf), CampusComputing, 2011 [↑](#footnote-ref-4)
4. <http://en.wikipedia.org/wiki/File:Software_Development_Spiral.svg> [↑](#footnote-ref-5)
5. <http://docs.moodle.org/23/en/Installing_Moodle> [↑](#footnote-ref-6)